Linux Journal Issue #126/October 2004

Features

Point-to-Point Linux *by Phil Hollenback*
When this Manhattan investment company decided to mirror its critical data off-site, the IS staff built their own T3 and T1 routers. How did that work, and would they do it again?

SQL Comes to Nmap: Power and Convenience *by Hasnain Atique*
Port-scan your own hosts to find misconfigured and unauthorized services. Put all that data into a database, and you can keep track of thousands of systems.

Setting Up Virtual Security Zones in a Linux Cluster *by Makan Pourzandi and Axelle Apvrille*
When projects need to share the Linux cluster but shouldn't see each other's data, split your in-demand cluster into separate virtual ones.

Indepth

Introduction to Sound Programming with ALSA *by Jeff Tranter*
The 2.6 kernel brings new capabilities to the Linux sound API. We cover the essentials with a working sound recording app.

The Politics of Porting *by Stephen C. Forster*
Don't do this. It could get you fired. Unless your company is really shooting itself in the foot, then you've got to do what you've got to do.

Linux Tools for Professional Photography *by RW Hawkins*

Tweak your system to make photo colors accurate, and more. Now you won't get a nasty surprise when the photo you send to *Linux Journal* shows up all wrong.

Embedded

Porting RTOS Device Drivers to Embedded Linux *by Bill Weinberg*
Your old real-time operating system made you do a lot for yourself as a driver author. Take advantage of the facilities Linux offers and clean up some spaghetti code while you're at it.

Toolbox

At the Forge Syndication with RSS *by Reuven M. Lerner*
Kernel Korner Filesystem Labeling in SELinux *by James Morris*
Cooking with Linux The Game of Security *by Marcel Gagné*
Paranoid Penguin Linux Filesystem Security, Part I *by Mick Bauer*

Columns

Linux for Suits Bridging the Gap *by Doc Searls*
EOF Dear Laptop Vendor *by Lincoln D. Durey*

Reviews

IBM's IntelliStation A Pro *by Chris DiBona*
Security Warrior *by Dan York*

Departments

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

Archive Index

Advanced search

LINUX JOURNAL

Point-to-Point Linux

Phil Hollenback

Issue #126, October 2004

This financial firm decided to build its own redundant WAN routers. Here's a no-nonsense look at the tricky parts and how it all worked.

Last summer, the investment company I work for decided to build a disaster recovery site. This location, 40 miles from New York City, would provide a mirror of our Downtown Manhattan operation. We decided to utilize Linux as much as possible for this project for the following reasons:

1. We primarily are a Linux operation already, so we could use our existing experience.
2. We could customize the configuration as much as we wanted, because everything would be open source.
3. We hoped Linux solutions would be less expensive than other solutions, such as Cisco's.

In this article, I focus on our use of Linux in wide-area network (WAN) routers. I define a WAN router as a system that connects to both wide-area links, such as T1 or T3 lines, and local-area networks, such as 100baseT, and forwards packets between the two networks.

Network Design

We purchased dedicated connections because this is a disaster recovery site and we need the connections to be as reliable as possible. Based on our calculations, one T3 (45Mb/sec) and four T1 (1.544Mb/sec each) lines would provide sufficient bandwidth for our operations. Ultimately, we decided to use the T3 link as the primary connection and leave the T1s as a bonded 5.7Mb/sec backup link.

The choice of WAN connectivity determined our network design. For redundancy, we installed two WAN routers at each site. The routers are

identical and contain hardware to connect to both the T1 and T3 links. With the use of splitter hardware, we hoped to connect all the WAN links to all the routers, as shown in Figure 1. However, that design ultimately turned out to be extremely difficult to implement, due to technical issues I discuss below.

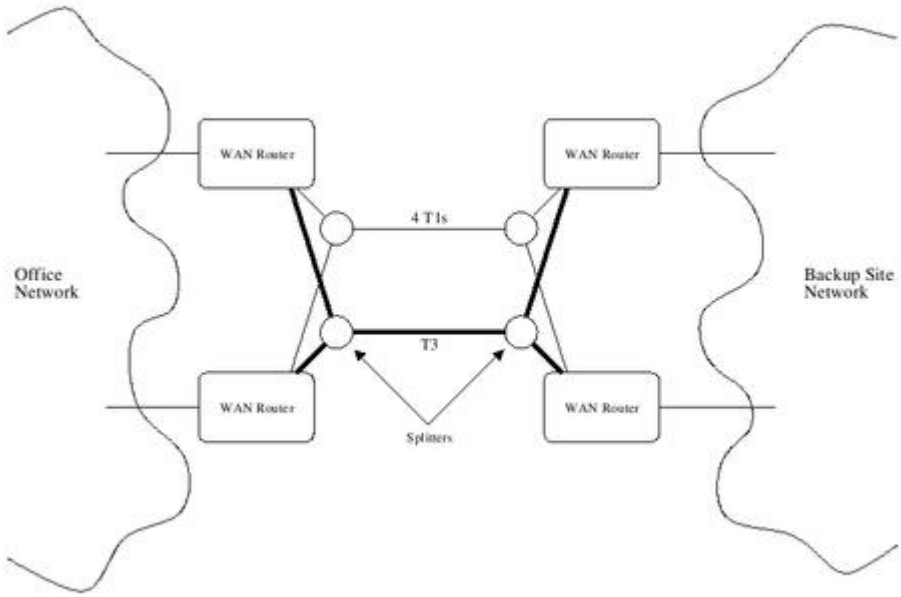


Figure 1. Redundant WAN Links

In addition to the WAN links, we also connected the remote site to the Internet through the hosting company backbone. We operated on the principle that more connectivity was better, and this turned out to be useful when we were designing the network. There's nothing like accidentally bringing down your T3 with a mistyped command to make you appreciate a back door to your routers over the Internet.

Hardware Considerations

Our space for servers at the hosting company was limited to one standard rack. This put space at a premium, because we needed to install a lot of servers. Thus, we decided to use 1U systems for the WAN routers. This was a difficult decision to make, as hardware options are limited in that form factor. In retrospect, it would have been much easier to use 2U systems for the WAN routers.

The next step was the selection of T1 and T3 interface cards. The main choice here is whether to use a card with an integrated channel service unit/data service unit (CSU/DSU) that connects directly to the incoming WAN circuit or a card with a high-speed serial connection along with a standalone CSU/DSU. Given our space constraints, an integrated card made the most sense. In

previous WAN installations, we used Cisco 2620 router boxes with T1 cards installed. However, that was not appropriate for this project because we wanted to connect multiple T1 and T3 lines.

After much searching, the only vendor we found that could supply both T3 and multiport T1 cards was SBE, Inc. The market for these cards is small and the number of vendors is limited. My suggestion for finding WAN cards is to start asking tech support a lot of questions and see how they respond. Also, carefully look over the driver and hardware specs before committing to a particular vendor.

Designing the Router Computers

With the T3 and four T1 cards from SBE, we would require a system with two free full-height, half-length PCI slots. We decided on Tyan S5102 motherboards with a single Pentium 4 Xeon 2.4GHz CPU. For memory, we used 256MB of ECC RAM for maximum reliability.

To cut down on the chance of system failure, we used Flash-based IDE devices. We found a device from SimpleTech that connects and operates like a conventional hard drive. We decided on a 256MB device as we thought that would be enough room for Fedora Core 1 to operate.

The complete computer systems (minus the WAN cards) were purchased from a white box system supplier. This proved troublesome, though, as the supplier was not able to produce four completely identical systems. The systems had variations in CPU fan manufacturers and memory speeds.

One area where the system supplier was helpful was in finding the right case. Only one of the numerous system vendors I contacted could supply a motherboard and case combination that could hold two full-height PCI cards. We had hoped to use a stock system from a supplier such as Dell or IBM, but none of the big names could give us a system that matched all our criteria.

Circuits and Cabling

It's critical to have redundant circuits connecting an office to a backup site. Determine who serves your sites and find a backup site served by multiple providers. Our office is connected physically to two providers, so we ended up ordering the T3 from one and the T1s from the other. If you don't research carefully which providers have actual physical connections to your sites, you are likely to end up with all your circuits running through one vendor's cable at some point.

T1s come on standard RJ-45 cables. Typically, the provider terminates the T1s—and their responsibility—at your demarcation point (demarc). The demarc generally is where all your phone connections are made. From the demarc, it is a simple matter to run regular Ethernet cables to your racks.

T3s are more complicated. The physical connection is two coaxial cables, one for transmit and one for receive. T3s use RG-59A cable with BNC connectors. The T3 provider informed us that our server room was too far from its equipment in our building, so a T3 repeater was necessary. This required 4U of space and a 120-volt outlet in our rack. Luckily, this distance flaw wasn't repeated at the hosting facility.

Splitting the Circuits

Our goal was to connect all circuits to all WAN routers (Figure 1) and leave the circuits turned off on the spare system on each end. One router at each end would be the master for the T3, and the other would be the master for the four T1s. If either router failed, the circuits could be brought up on the other router.

Based on our research, in particular, some of Cisco's high-end telco equipment, we knew that splitting the circuits was possible. The key constraint is only one system on each end can be transmitting and receiving at a time. That turned out to be a large problem because SBE's hardware was not designed to be inactive while connected to a line. The critical flaw was the transmitter on the T3 cards automatically turns on when power is applied to the card. So, if you have the T3 circuit up running between two systems, one on each end, and you power-cycle the spare system on one end, the T3 goes down because both systems on one end are trying to transmit. This can be worked around partially by sending a shutoff command to the transmitter on the card. This isn't possible until the machine is loaded and the OS is installed, a potential delay of several minutes.

We also discovered that the T3 signals on the coaxial cables must be impedance-matched. The impedance on a T3 cable is 75 ohms. If you simply split that connection, the impedance on the two resulting cables is 37.5 ohms, which may or may not work, depending on your hardware. The correct way to split T3 cables is to use what's called a power splitter, which contains a transformer to balance the impedance properly at 75 ohms on all connections. We used passive power splitters from Micro Circuits, Inc.

Splitting the T1s was much simpler. It's sufficient to use RJ-45 tee connectors to turn one incoming cable into two outgoing cables. Also, the SBE 4T1 card is designed to not turn on the transmitter until the driver is loaded, so you can share the connection between systems.

We were able to make all these split connections work. However, due to the startup problems with the T3 cards and other issues, we currently do not have the splitters installed. If you want to try doing this step, you have to get everything working rock solid without splitting before even attempting it. Otherwise, you will be removing the splitters every time there is a problem with a circuit because you won't have confidence in your setup.

Rolling Our Own Distro

The choice of a 256MB Flash drive for storage dictated a compact OS install. At Telemetry, we have standardized on Fedora Core 1 for all Linux systems. Thus, it was convenient to run FC1 on the router systems as well. The two goals:

1. Create something similar to stock Fedora Core 1 that would fit on a small drive.
2. Change the system configuration to avoid unnecessary writes to the drive. This is important because Flash drives have a finite lifetime, so placing log files on them is a bad idea.

It turns out to be relatively easy to build a custom Fedora system, especially compared to what was available in previous Red Hat releases. The key is to build your own system image on another machine with a fresh RPM database and then transfer that image to the router. Listing 1, available from the *Linux Journal* FTP site [ftp.linuxjournal.com/pub/lj/listings/issue126/7661.tgz], shows how to build a basic system image. The procedure is to create a new RPM database somewhere on your build system, install a minimal set of RPMs to create the system and then install all other RPMs you want. I use the --aid option to rpm to tell it to satisfy all dependencies automatically by looking in a directory where I have placed copies of all the Fedora RPMs. This saves me the work of manually determining all the dependencies. Once you have the system image built, copy it over to the router for testing. We were able to create a workable system that used 171 of the 256MB available on the Flash drive.

Tweaking the Router Configuration

The goal of using a Linux router is to minimize disk reads and writes. This is necessary because the memory in a Flash drive can be written to only a fixed number of times, typically in the hundreds of thousands. The way to minimize writes is to treat the router as though it were a laptop. First, enable laptop mode in the kernel, as described in the September 2004 issue of *Linux Journal*. This causes the system to delay writes until a read is requested instead of sending writes to the disk as soon as they occur.

Second, adjust your filesystem mounting options to delay writes as well. For ext3, set the commit interval to 60 seconds. Then, mount the filesystem with

the noatime option so that reads on files don't generate a write of modified access times.

Third, move all log files off the drive and into a RAM-based filesystem, tmpfs. Listing 2 shows how to restructure your filesystem to move all log files out of /var and into a tmpfs called /var/impermanent. For this to be really useful, you also need a script, such as the one in Listing 3, that saves all the log files in a tarball on system shutdown and restores them on boot [Listings 2 and 3 also are available on the *LJ* FTP site]. This script should be called as early as possible in /etc/rc.d/rc.sysinit on startup and as late as possible during shutdown in /etc/init.d/halt.

Configuring the WAN Links

WAN links are confusing! For example, the T3 and T1 drivers use different versions of the kernel HDLC stack. This means we have to keep two different versions of the sethdlc program used to set the protocol on the WAN circuit, one built against each hdlc stack.

There are many configuration parameters to set on a T3 or T1 circuit—external or internal timing? CRC size? HDLC mode? and so on. Fortunately, SBE's tech support was helpful and supplied many configuration and troubleshooting tips.

We decided to bond the four T1s into one bonded circuit, using teql. This worked, but performance was terrible if one of the T1s was removed, even after it was reconnected. My coworker, Bill Rugolsky, tracked the problem down to a lack of link status reporting. The SBE card could report whether the link was up or down, but this message was not propagated up the stack. Thus, teql continued to try to send out packets using down interfaces. Bill resolved this by patching the SBE driver and installing patches others had created to fix teql and linkwatch notification. The driver patches were provided to SBE, and we hope they are included in the next revision of their driver.

Our boss Andy Schorr did the work to set up OSPF to handle routing over the WAN links. The open-source package Quagga, a successor to Zebra, provides the necessary framework. If one of the links goes down (remember, there are two links, the T3 and the virtual link over the bonded T1s), Quagga detects this and starts routing packets over the other interface. Traditionally, point-to-point links are configured to borrow the address of another interface, typically eth0. However, we decided to use dedicated subnets for each point-to-point link. Andy had to modify the source code to make Quagga work properly in this setup.

We also had to figure out some iptables rules to make Quagga work correctly with the bonded T1s. The teql device is send-only, so packets never appear on

it. This causes Quagga to drop the packets, because they come in on the wrong interface. The fix is a couple of iptables rules to make packets arriving on all the T1 interfaces (hdlc0 through hdlc3) appear on teql0:

```
iptables -t mangle -A PREROUTING -i hdlc\+ -j TTL --ttl-inc 1
iptables -t mangle -A PREROUTING -i hdlc\+ -j ROUTE --iif teql0
```

The bottom line is that setting up WAN links is tricky work and requires much study and tweaking. Don't expect things to work simply because you connect the cables.

Obstacles along the Way

We had to resolve a number of problems while configuring these WAN routers. Some of the earlier ones were with the WAN drivers, as mentioned above. As I was writing this article, we discovered that our T1 performance had deteriorated badly, with highly variable ping times—up to 1 second instead of the usual 10ms. We traced this to one of the WAN cards not generating interrupts; it had come loose in the PCI slot. The widely varying packet delays were occurring because the other device sharing the same interrupt line (eth0) was sending interrupts. This, in turn, caused the SBE driver to wake up and process its interrupts. This type of non-obvious failure highlights the importance of link-quality monitoring.

The Future

We are satisfied with the basic architecture, but a number of improvements need to be made. Given the annoyances of managing multiple T1s in a bonded interface, we now are planning on upgrading the T1s to a second T3. When we do that, we may drop the circuit splitting entirely. Circuit splitting adds a whole new level of complexity to the entire system, and we are unsure if it is worth it.

We have to continue to improve our monitoring of both line status and line quality. It is difficult to complain to circuit vendors about performance if you don't have historical data to back up your assertions.

It would have been convenient to use off-the-shelf servers for the router boxes. We have been investigating the latest 1U rackmount from a major manufacturer, but for several reasons it is unsuitable. The showstopper is that the BIOS does not allow booting from any Flash IDE device. The vendor knows of this limitation but will not fix the BIOS. Thus, we see ourselves building our own systems for the foreseeable future.

We will be building additional internal router boxes for handling LAN traffic, based on the WAN router model we have developed—1U systems with Flash drives running a minimal Fedora kernel.

Conclusion

Although this project is not complete, I feel we've accomplished enough to take a moment to evaluate its success. The key question is: would we do it again? The answer is a qualified yes. Our WAN routers perform the task of providing redundant connections between our office and backup sites. The usefulness of splitting the WAN circuits for redundancy is a wash as it adds so much complexity to the design.

This project has taken significantly longer to complete than we anticipated, a general symptom of developing your own solutions. The answers are there, but you expend more time finding them. Having a sharp, dedicated team (as I did) is crucial to making it all work. Just make sure to budget extra time for all the annoying little problems that are sure to arise.

Resources for this article: /article/7703.

Phil Hollenback is a system administrator at Telemetry Investments in New York City. When he's not upgrading Linux servers or skateboarding, Phil spends his time updating his Web site, www.hollenback.net.

Archive Index Issue Table of Contents

Advanced search

SQL Comes to Nmap: Power and Convenience

Hasnain Atique

Issue #126, October 2004

When you're using Nmap to check the security of many hosts, put MySQL to work keeping track of trends and changes.

I recently was exchanging e-mail with someone who regularly needs to port scan his own network for vulnerability trending. The port scanning tool of choice for this task is Nmap, but managing the data from Nmap was an entirely different beast. A few weeks later, a patch to Nmap that allows it to log the results directly to MySQL was ready. Although Nmap supports machine-parseable format as well as XML output, the ability to log directly to an SQL database far outruns XML or even machine-parseable output. For one, nmapsql does not involve an extra step in the shell to feed the output to a back end.

nmapsql is a direct patch applied to Fyodor's venerable Nmap v3.48 port scanning tool (at the time of this writing, Nmap v3.50 had just been released; an updated version of nmapsql for v3.50 is available from the Web site). It adds MySQL support, but it goes beyond merely adding the results; it also does target tagging, scanner tagging and simple trending. Once the data has been captured in an SQL database, a whole new set of tasks is possible. nmapsql can be downloaded from sourceforge.net/projects/nmapsql. At the moment, it relies on MySQL's client interface for data manipulation.

As security administrators aren't necessarily database wizards, nmapsql was designed to be simple to use. It's simple enough that most of the information one might want in a network scan can be obtained from a single table. Simplicity is also why IP addresses are stored as plain text instead of with inet_aton() notation. I'm aware of the performance penalties of text manipulation, but the focus is to demonstrate the convenience with a small data set. The target tags, runtime and scanner IDs are there for numeric searches in large data sets where performance is critical.

In this article, we concentrate first on running an SQL-enabled scan to establish a baseline of open ports and live targets on a network. Later, we take a look at the data captured in SQL and find ways of comparing the results.

The Options

nmapsql starts out by reading the ~/nmpsql.rc file in the effective user's home directory. So, if you used `su` to get to root before running nmapsql, ~root/nmapsql.rc is read. At this time, only four items are read from nmapsql.rc, each on a line by itself and in the item=value format common to many other utilities. The items are server=localhost, db=nmaplog, user=nmap and passwd=scanamanga.

The server is the DNS name of the host where MySQL is running, and db is the name of the database on that server. The user and password items are used to connect to the database, and the user listed must have at least SELECT, INSERT and UPDATE rights to the database.

On the command line, nmapsql introduces four new options to those Nmap already provides: --mysql, --runid, --targetid and --scannerid. When the nmapsql binary is executed without any of these options, it behaves exactly as normal Nmap does. None of these options interfere with Nmap's existing output abilities, so it's entirely possible to log to SQL as well as to produce machine-parseable output from the same scan.

The --mysql option, without any of the other nmapsql options on the command line, triggers MySQL logging, with all tags and IDs auto-assigned. All other nmapsql options automatically assume --mysql. Auto-assignment always picks the maximum available value in the respective table and increments by one.

The scanner ID feature, initiated by the --scanner-id *xxx* option, where *xxx* is the ID value, is intended for scenarios where more than one scanner is deployed, perhaps in a multisubnet environment. The scanner ID, along with the runtime ID, is stored in the portstat table to allow separation of result sets by the scanning host. It would be simple to separate the results of scanner ten, for instance, using a query like this:

```
mysql> select * from portstat
    -> where scannerid = 10 and runid = 100;
```

The --run-id *xxx* option is used to specify a specific ID for the current nmapsql run. If this option is not specified, a system-generated ID is used. If the runid specified already exists in the database, it is reused. This feature allows results of multiple scans to be grouped conveniently under a single runid.

The runtime ID and its associated information are stored in the runlist table. See the “Tables Used by nmaplog” sidebar for a summary of the tables used. Some of the runtime information is updated at the end of the scan, including the total number of possible targets specified on the command line and the total number found alive. Similarly, the scanner ID and related information go to the scanners table.

Tables Used by nmaplog

At this time, eight tables can be found in the nmaplog database. Relevant tables are:

- TARGETS—contains information about the target host, including the target ID, IP address, resolved hostname and OS guessed by Nmap. The hostid field links to the portstat table.
- SCANNERS—contains information about the host on which nmapsql is executed. The scan_id field is our link to the portstat table.
- RUNLIST—contains the user ID, date and time information about each invocation of Nmap, including the host from which it was run. The user name and user ID are from /etc/passwd. The scanner_id field ties to scanners.scan_id.
- PORTSTAT—contains the port scan results. Each port reported by nmaplog is recorded, along with a state (open/close/filtered). Nearly all of the other tables link to this table through the ID fields.
- HOSTSSTAT—contains rudimentary statistics about the target host for each run of nmapsql, such as the total ports scanned and number of ports found open.

Each target scanned also is assigned a tag and the information stored in the targets table. As with the runlist, the rows in the targets table are populated in two stages. The first stage captures the IP address and the hostname if resolvable, and the second stage populates the os_guessed column. At this time, the fingerprint information for unrecognized OS is not stored, but it may be in the future. No duplicates ever are created in the targets table. In my experience, the only situation where you might have duplicate IP subnets is when you move from one customer to another. A different database for each customer should be used in such cases.

The target IDs are not used at the moment, but you're able to specify your own target ID for any target on the command line. If the specified ID exists, it is ignored and a system-generated ID is used instead, for the sake of uniqueness. If the target ID value after --target-id on the command line does not exist in the targets table, it is assigned to the IP address of the current target. If the target

specification is for multiple systems, the first target has the specified target ID, with the subsequent ones being assigned incremental IDs.

The Basics

nmapsql logs the date and time of execution, the user who executed Nmap, the host on which Nmap is running and an identification number for the execution. These last two items allow nmapsql to be used in large environments and form the basis of comparison among scans. The runid, or runtime ID, is always unique within that data set. If the target specification remains the same, the runid alone can differentiate the results of two scans. But it's also possible to group results of multiple scans under a single runid using the --run-id command-line option. For instance, consider the following invocation of nmapsql:

```
$ nmap -A --mysql --runid 100 192.168.10.1/24
```

This command starts Nmap with the logging functionality enabled by the --mysql option, assigns 100 for the current runid and scans the 192.168.10.1/24 network. If this is the first invocation of nmapsql, this would establish a baseline for the network against which all subsequent runs could be compared. nmapsql also automatically creates an entry for the host on which it's running, in this case 192.168.10.44, and assigns it a scanner_id in the scanners table. Partial console output from Nmap for this run is shown in Listing 1.

Listing 1. Partial Output from Nmap

```
Starting nmap 3.48 ( http://www.insecure.org/nmap/ )
at 2003-12-14 10:00 SGT
Insufficient responses for TCP sequencing (1),
OS detection may be less accurate
Interesting ports on 192.168.10.0:
(The 1656 ports scanned but not shown below are in
state: closed)
PORT   STATE SERVICE VERSION
80/tcp open  http?
Device type: print server
Running: Linksys embedded
OS details: Linksys EtherFast print server

Interesting ports on wap.hasnains.com (192.168.10.1):
(The 1654 ports scanned but not shown below are in
state: closed)
PORT   STATE SERVICE VERSION
21/tcp open  ftp?
23/tcp open  telnet?
80/tcp open  http
Device type: broadband router
Running: Zyxel ZyNOS
OS details: ZyXEL Prestige 700/Netgear MA314
broadband router
```

The target specification in this example is the entire Class C subnet. nmapsql auto-assigns a unique target ID for each live host in the network and stores

additional information in the hoststats table. This table alone can be a poor-man's port scan result comparison tool.

Let's take a quick look at what was logged. To do that, we log in to the MySQL client and connect to the database listed in the nmapsql.rc file. Then we issue the query:

```
$ mysql nmaplog -p
mysql> select target_ip, d, t, port, protocol,
    -> state, runid from portstat
    -> order by target_ip, d, t ;
```

This query would produce the table shown in Listing 2. It provides a nice listing ordered by target IP, date and time. Notice that the runid column has 100 for all the rows as stated on the command line.

Listing 2. Results of a Single Scan

```
+---------------+------------+----------+--------+----------+--------+-------+
| target_ip     | d          | t        | port   | protocol | state  | runid |
+---------------+------------+----------+--------+----------+--------+-------+
| 192.168.10.0  | 2003-12-14 | 10:00:37 |     80 | tcp      | open   |   100 |
| 192.168.10.1  | 2003-12-14 | 10:00:37 |     21 | tcp      | open   |   100 |
| 192.168.10.1  | 2003-12-14 | 10:00:37 |     23 | tcp      | open   |   100 |
| 192.168.10.1  | 2003-12-14 | 10:00:37 |     80 | tcp      | open   |   100 |
| 192.168.10.44 | 2003-12-14 | 10:00:37 |     22 | tcp      | open   |   100 |
| 192.168.10.44 | 2003-12-14 | 10:00:37 |    111 | tcp      | open   |   100 |
| 192.168.10.44 | 2003-12-14 | 10:00:37 |   3306 | tcp      | open   |   100 |
| 192.168.10.44 | 2003-12-14 | 10:00:37 |   6000 | tcp      | open   |   100 |
| 192.168.10.64 | 2003-12-14 | 10:00:37 |    135 | tcp      | open   |   100 |
| 192.168.10.64 | 2003-12-14 | 10:00:37 |    139 | tcp      | open   |   100 |
| 192.168.10.64 | 2003-12-14 | 10:00:37 |    445 | tcp      | open   |   100 |
| 192.168.10.64 | 2003-12-14 | 10:00:37 |   1024 | tcp      | open   |   100 |
| 192.168.10.64 | 2003-12-14 | 10:00:37 |   1025 | tcp      | open   |   100 |
| 192.168.10.64 | 2003-12-14 | 10:00:37 |   1031 | tcp      | open   |   100 |
| 192.168.10.64 | 2003-12-14 | 10:00:37 |   5000 | tcp      | open   |   100 |
| 192.168.10.64 | 2003-12-14 | 10:00:37 |   5101 | tcp      | open   |   100 |
| 192.168.10.64 | 2003-12-14 | 10:00:37 |   6000 | tcp      | open   |   100 |
| 192.168.10.65 | 2003-12-14 | 10:00:37 |    135 | tcp      | open   |   100 |
| 192.168.10.65 | 2003-12-14 | 10:00:37 |    139 | tcp      | open   |   100 |
| 192.168.10.65 | 2003-12-14 | 10:00:37 |    445 | tcp      | open   |   100 |
| 192.168.10.65 | 2003-12-14 | 10:00:37 |   1025 | tcp      | open   |   100 |
| 192.168.10.65 | 2003-12-14 | 10:00:37 |   5000 | tcp      | open   |   100 |
+---------------+------------+----------+--------+----------+--------+-------+
```

We get Listing 3, showing the open ports information for target 192.168.10.44, when we use the following query:

```
mysql> select target_ip, d, t, port, protocol,
    -> state, runid from portstat
    -> where target_ip = '192.168.10.44'
    -> order by d, t;
```

Listing 3. Scan Results for a Single Host

```
+---------------+------------+----------+--------+----------+--------+-------+
| target_ip     | d          | t        | port   | protocol | state  | runid |
+---------------+------------+----------+--------+----------+--------+-------+
```

```
| 192.168.10.44 | 2003-12-14 | 10:00:37 |     22 | tcp      | open   |   100 |
| 192.168.10.44 | 2003-12-14 | 10:00:37 |    111 | tcp      | open   |   100 |
| 192.168.10.44 | 2003-12-14 | 10:00:37 |   3306 | tcp      | open   |   100 |
| 192.168.10.44 | 2003-12-14 | 10:00:37 |   6000 | tcp      | open   |   100 |
+---------------+------------+----------+--------+----------+--------+-------+
```

If you match the four lines of output with the section for 192.168.10.44 in Listing 1, you can see the relationship immediately. As shown here, the portstat table alone can provide all the port scan information from Nmap. Of course, if you've done a number of scans, the above query shows all the results for 192.168.10.44 found in all the scans.

Let's say that two days, or a week or a month later, you scan your network again and want to compare the two results visually. A quick look through the runlist table shows that runid 102 corresponds to two days after the first scan. Armed with that information, you enter the query:

```
mysql> select target_ip, d,t,port, protocol,
    -> state from portstat
    -> where target_ip = '192.168.10.44'
    -> and runid = 102 order by d,t,port;
```

You easily can compare the results of Listings 4 and 5 to pick out the differences. Obviously, a program could compare the two result sets and summarize the differences for you.

Listing 4. 192.168.10.44 Scanned Two Days Later

```
+---------------+------------+----------+--------+----------+--------+
| target_ip     | d          | t        | port   | protocol | state  |
+---------------+------------+----------+--------+----------+--------+
| 192.168.10.44 | 2003-12-16 | 00:47:16 |     22 | tcp      | open   |
| 192.168.10.44 | 2003-12-16 | 00:47:16 |    111 | tcp      | open   |
| 192.168.10.44 | 2003-12-16 | 00:47:16 |   3306 | tcp      | open   |
| 192.168.10.44 | 2003-12-16 | 00:47:16 |   6000 | tcp      | open   |
+---------------+------------+----------+--------+----------+--------+
```

Listing 5. Combined Results for a Host

```
+------------+----------+---------------+----------------------+------------------------------+--------
| d          | t        | ip            | host                 | os_guessed                   | port
+------------+----------+---------------+----------------------+------------------------------+--------
| 2003-12-14 | 10:00:37 | 192.168.10.44 | ophelia.hasnains.com | Linux Kernel 2.4.0 - 2.5.20 |     22
| 2003-12-14 | 10:00:37 | 192.168.10.44 | ophelia.hasnains.com | Linux Kernel 2.4.0 - 2.5.20 |    111
| 2003-12-14 | 10:00:37 | 192.168.10.44 | ophelia.hasnains.com | Linux Kernel 2.4.0 - 2.5.20 |   3306
| 2003-12-14 | 10:00:37 | 192.168.10.44 | ophelia.hasnains.com | Linux Kernel 2.4.0 - 2.5.20 |   6000
+------------+----------+---------------+----------------------+------------------------------+--------
```

Going back to the poor man's port scan results, the hoststat table contains information for each live host. It keeps a simple count of open ports in the open_ports column. To find the open port information for our target host, we query:

```
mysql> select ip,d,t,open_ports, ports_scanned,
    -> runid from hoststats where order by ip, d,t;
```

to receive a single line of output. (We added the order by clause for future use.) The open_ports column, when viewed along with the date/time and runid columns, sketches a rough trend of open ports over a period of time.

The targets table captures information on each target it encounters, one row per unique IP address. This is the only place where the hostname, if resolvable, and the OS guessed by Nmap are captured. Let's find out what it knows about our target:

```
mysql> select  * from targets
    -> where ip = '192.168.10.44';
```

Notice that the OS_guessed field now contains Linux Kernel 2.4.0-2.5.20 and the hostname column is set to ophelia.hasnains.com (I like Shakespeare's tragic heroines).

Now that we basically have all the bits and pieces, let's construct a single query to put all the information in one place for our target host:

```
mysql> select r.runid, r.d, r.t, t.ip, t.host,
    -> t.os_guessed, p.port, p.protocol, p.service,
    -> p.state, p.fullversion from runlist r,
    -> targets t, portstat p
    -> where r.runid = 100 and p.target_ip = t.ip
    -> and p.runid = r.runid
    -> order by r.runid, r.d, r.t, t.ip;
```

We're not showing the output for reasons of brevity, but you could try it on your own. We could use a report writer to group the results by targets. For fancier output, we need to get heavier artillery, such as PHP or Perl. One of the most useful reports is to identify the change in open ports for each target. For instance, say our target has closed 111/TCP but opened 23/TCP. In such a scenario, the open_ports column in hoststats still would show four ports even though the details have changed. But a custom program easily could pick out the difference(s) to report.

Useful Queries

The most common query would be to find out what ports are open for a given target, and that can be accomplished with:

```
mysql> select d, t, port, protocol, state,
    -> fullversion from portstat
    -> where target_ip =  '192.168.10.44'
    -> order by d,t,port;
```

Another common query is whether a given port was open on a target at some time in the past—"Did we have SSH open on 192.168.10.44 two weeks ago?" As long as nmapsql was installed, assuming nmapsql runs routinely from crontab, the answer would be in the following query:

```
mysql> select d, t,target_ip, port,protocol,
    -> service, state, fullversion from portstat
    -> where port = 22 and protocol = "tcp"
    -> and state = "open"
    -> d = date_sub( curdate(), interval 14 day)
    -> order by d, runid, target_ip ;
```

Obviously, you could have more than one instance of nmapsql running on a given day, hence the order by clause. If you were using a third-party tool, such as PHP or Perl, to generate the result set, you could consult the runlist table to find the runid for the exact time frame you need and query for results of that runid for your target of choice.

Another useful query is to identify the total number of targets in a given network with a given port open—"How many hosts in the 192.168.10/24 subnet have 80/TCP open?" This query would produce the following result:

```
mysql> select runid, d, t, target_ip, port,
    -> protocol, state from portstats
    -> where port = 80 and protocol = 'tcp'
    -> and state = 'open'
    -> and target_ip like '192.168.10.%';
```

Text matching doesn't quite lend itself to subnet matching, but you get the general idea.

Using a Different Database

In many cases, such as when a consultant goes from one network to another, it's desirable to be able to change the name of the database, perhaps to the customer's name, so that the data from multiple places doesn't get jumbled together. At the time of this writing, the way to do that is to update the db=nmaplog item in the ~/nmapsql.rc file used by nmaplog to pick up database access information.

In order to change the database in use, replace nmaplog in ~/nmapsql.rc with the appropriate name, and then make sure the user specified in ~/nmapsql.rc has permissions on that database. Then, load the database schema into the new database. Assuming the new database is called newnmap, the following line would load the schema:

```
$ mysql newnmap < nmaplog.sql
```

I don't recommend using different databases, however. It's far easier to unload the data to a disk file and then load a blank schema into the nmaplog database. The following lines would accomplish this:

```
$ mysqldump nmaplog > newnmap.sql
$ mysql nmaplog < nmaplog.sql
```

Depending on how your database permissions are set up, you might have to specify the MySQL user name and password for the above commands to work.

Deployment

nmapsql's usefulness is hard to appreciate when run infrequently on one or two targets. It's in large environments with multiple subnets and dozens of targets where nmapsql really shines. The simplest deployment, of course, is where nmapsql and the MySQL server reside on the same host, such as a laptop a consultant carries from network to network. Because most networks are firewalled and use RFC 1918 addressing, duplicate IP addresses in the targets table is highly possible with a single laptop in roving environments. In these cases, you should unload the data and use a fresh database for each new environment.

nmapsql lends itself to other deployment scenarios: mid-sized environments where multiple scanners from different subnets log back to a single MySQL server and large environments where multiple self-contained (MySQL and nmapsql on the same box) systems do their local scanning and logging. In both these environments, duplicate RFC 1918 addresses are unlikely. However, because of the lag between scanning/logging locally and collecting to the central server, the data isn't in real time. These are two situations where the scanner ID is useful to separate data.

Future Directions

Security practitioners—and I must admit, some black hats—appreciate nmapsql's functionality, as it fulfills a great need. The project's immediate goals are to allow users to set nmapsql-specific options from inside nmapfe, the Nmap front end, and to build a reporting front end with PHP so end users do not have to enter queries manually in MySQL. Both of these currently are under development.

Looking further, there are plans to integrate the results of Nessus vulnerability scans into the same database, creating a single console for port scan vulnerability assessment results. Toward that goal, nmapsql's Web site

currently has a simple parser that loads result files created from the Nessus client.

Hasnain Atique (hatique@hasnains.com) lives in sunny Singapore with his wife and three-year-old daughter. When he's not watching *Harry Potter* with his daughter, he tries to be the lord of the pings and occasionally succeeds.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Setting Up Virtual Security Zones in a Linux Cluster

Makan Pourzandi

Axelle Apvrille

Issue #126, October 2004

The Distributed Security Infrastructure lets you create disjoint virtual security zones on a physical Linux cluster.

An increasing number of projects use Linux and other open-source software as basic building blocks to create clusters. Examples range from clusters that perform massive computations of visual effects for movies to clusters used as next-generation telecommunication servers.

More and more often, various issues, including economics, management and flexibility, require applications to run on the same physical cluster. An illustration of this situation in the telecom world is carrier-grade clustered servers shared among different operators. The operators share the global infrastructure of the cluster and provide different services to their clients but want to keep their binaries and data private. In such cases, cluster administrators do not have access to the source code for these applications, and security mechanisms cannot be enforced at the source code level. Hence, a security infrastructure is needed to ensure that a given application's resources cannot be tampered with or used by other entities on the cluster.

The Distributed Security Infrastructure (DSI) provides a solution for such a situation. It attempts to build a coherent security framework dedicated to carrier-grade Linux clusters by dividing a cluster into several virtual subclusters, guaranteeing controlled/restricted connections between them. Even though the project is only in its second year of design and development, we believe DSI is a useful tool for cluster administrators. This article presents how to use DSI to set virtual security zones inside a Linux cluster.

DSI Architecture and DSI Tools

In this section, we briefly introduce DSI's architecture. DSI is composed of one security server (SS) and multiple security managers (SMs), one per node (Figure 1). The SS centralizes management of the cluster: it gathers alarms and warnings sent by SMs and propagates a unique security policy over the cluster. On the other hand, SMs are responsible for enforcing security on their own nodes. Furthermore, messages are exchanged between the SS and SMs over encrypted and authenticated channels, using SSL/TLS over CORBA event channels.

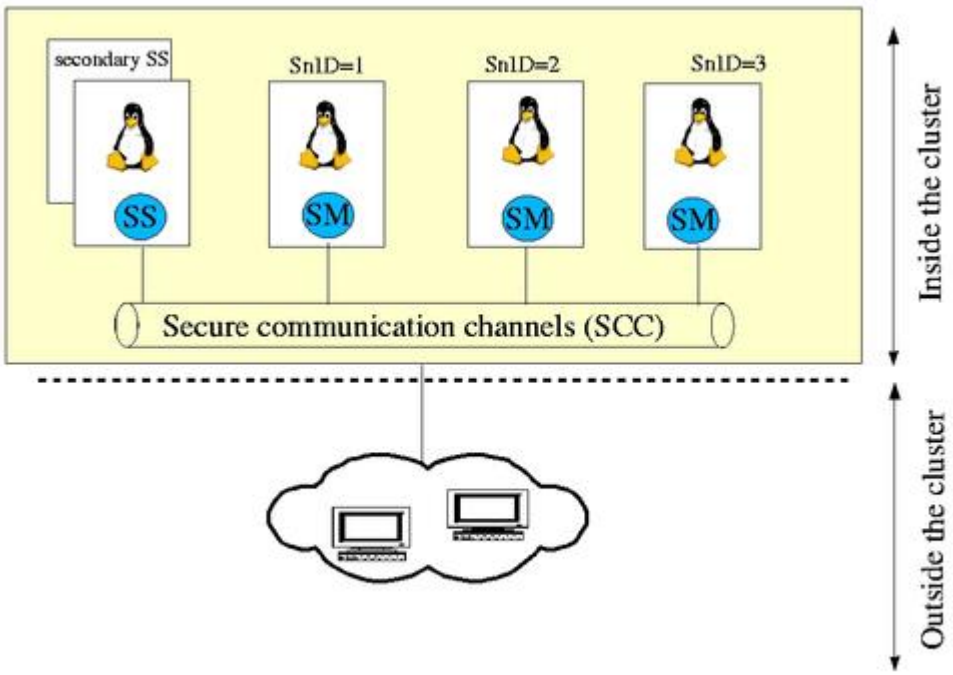


Figure 1. The DSI Architecture

Security mechanisms in DSI are implemented at the process level to check the access privileges a process has to a resource. Each process is identified by its security context identifier (ScID) and the node identification on which it is running (SnID).

SnIDs are assigned using the DSI SetNodeID tool. All processes sharing the same security context are assigned the same ScID. ScIDs can be assigned automatically by the system according to DSP rules (see below), or they can be assigned specifically to a given binary using the DSI SetSID tool. This allows grouping of binaries according to their security contexts.

The DSP Configuration File

In DSI, writing a security policy for the cluster consists of granting or denying permissions to a given SnID and ScID pair. These rules are valid for the whole cluster. All rules are centralized in an XML file on the SS to ease management.

DSI provides a way to update and enforce transparently and automatically a unique homogeneous view of the whole cluster's security. Once the administrator modifies existing rules or adds a new rule to the distributed security policy (DSP), the DSP must be loaded on the SS using the dsiUpdatePolicy tool. Then, dsiUpdatePolicy checks the DSP file against our DSP XML schema (syntactical checks). If the DSP is validated, the SS propagates the new rules to all nodes of the cluster using the secure communication channels. Finally, each SM enforces the rules at kernel level calling the distributed security module (DSM, see Figure 2). DSM is based on the LSM kernel patch. Its detailed description is beyond the scope of this article; see the on-line Resources section for links to more information.

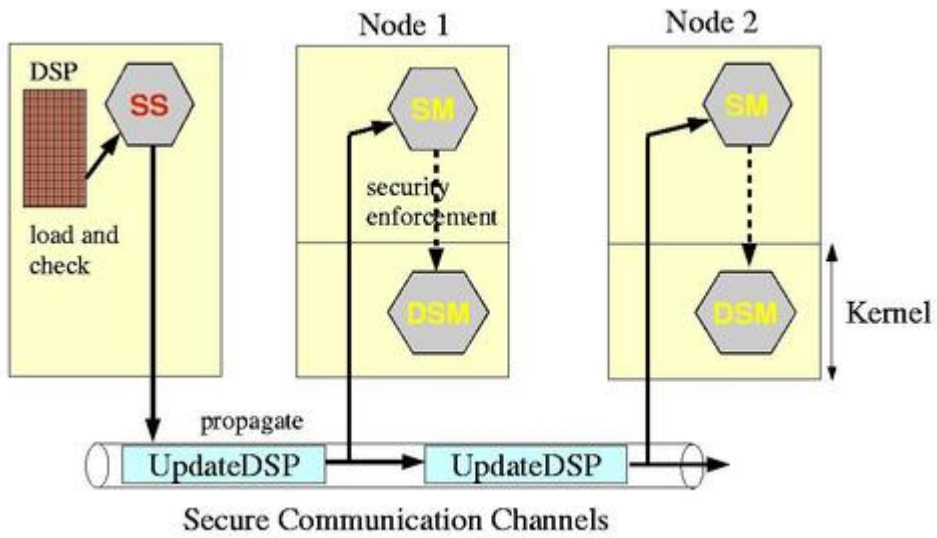


Figure 2. DSP Propagation inside the Cluster

Distributed Access Control

Controlling access to local resources is rather simple: the DSM module retrieves the local ScID and SnID of the requesting process and checks corresponding permissions in the security rules. Actually, the originality of DSI lies in distributed access control. Currently, only socket communications are

implemented. To illustrate this, we detail the access control mechanisms when a process tries to access a resource located on another node (Figure 3):

- The access request first is intercepted by the local DSM, which checks that the process has the privilege to call locally the socket-related systems calls.
- Then, the ScID and SnID of the requesting process are added by DSM to each IP packet sent to the remote node.
- On the receiving node, the remote DSM uses the ScID and SnID of the requesting process, extracted from the IP packet, to check its permission to communicate with both the target socket and the process to which the target socket belongs.
- Finally, the remote DSM locally checks that the process to which the target socket belongs may receive information from the requesting process.

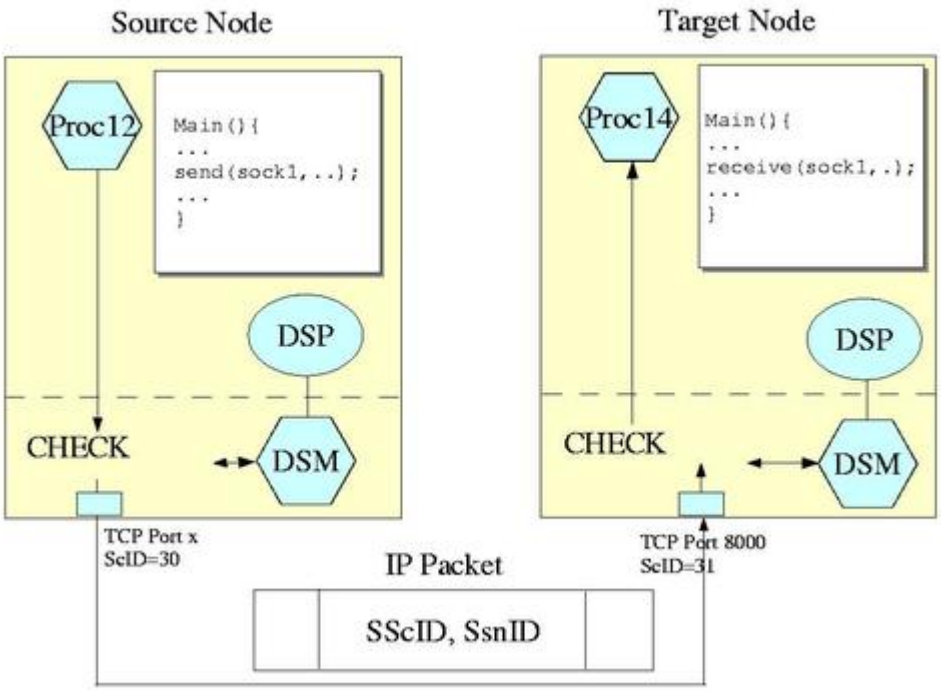


Figure 3. Secure Remote Access Control

The Problem

In this section, we walk through a simple scenario, which presents a problem and explains how DSI can help solve it. Say we want to share a cluster of two nodes (we begin small), among two telecommunication operators, PhoneMania and RingBell, each running their own applications on the cluster's nodes. Both offer a phone quotation service: end users call entry point servers (using TelecomClient) and request quotes for given companies. The entry point servers (PhoneManiaEP and RingBellEP) forward the requests to their back-end

servers (PhoneManiaBE and RingBellBE), which retrieve the quotes and send them back to the end user.

From a cluster operational point of view, the problem is the following: how can we prevent a PhoneMania application from forwarding its requests to RingBell's back-end servers? Without any specific security infrastructure, PhoneMania could do so when its back-end server is overloaded or simply when it does not have the requested information—not to mention more aggravated scenarios of subscriber's data theft or intentional harm meant for competitors and so on. To illustrate such a scenario, we implemented all actors as simple UDP client and server applications (Figure 4).

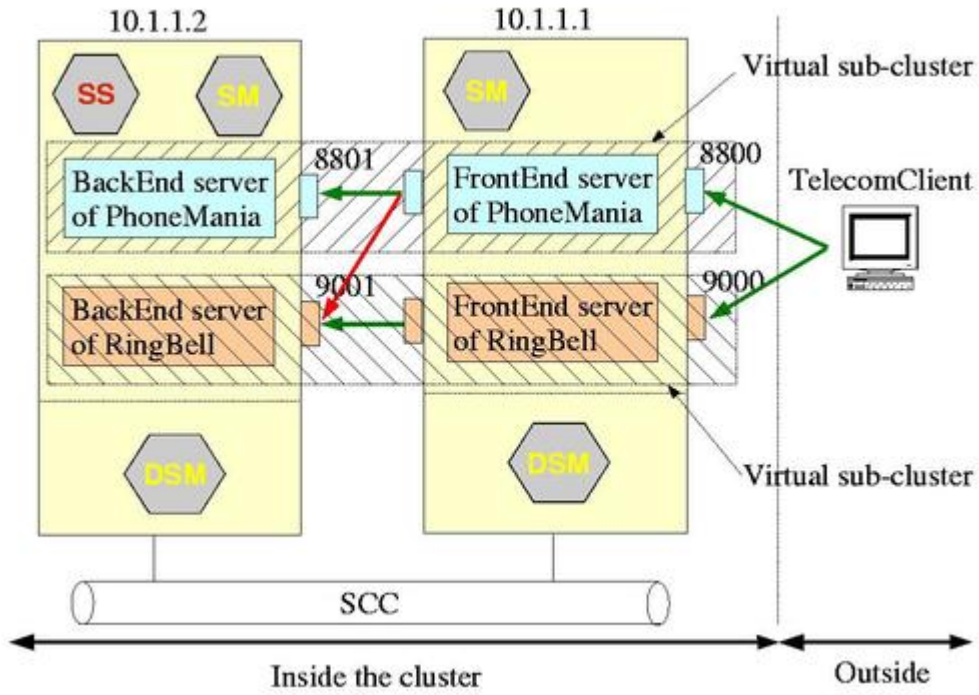


Figure 4. Simple Telecom Scenario

Here is the fraud scenario step by step:

- PhoneMania and RingBell launch their back-end servers on a node called munster:

```
[munster demo]$ ./RingBellBE -h 10.1.1.2 -p 9001
RINGBELL: bind on 10.1.1.2:9001
..
[munster demo]$ ./PhoneManiaBE -h 10.1.1.2
-p 8801
PHONEMANIA: bind on 10.1.1.2:8801
```

- Then, as PhoneMania is overloaded, he decides to use RingBell's resources. So, on node colby, the entry point server of PhoneMania (port 8800) forwards all requests from his customers to RingBell's back-end servers (port 9001):

```
[colby demo]$ ./PhoneManiaEP -h 10.1.1.1
-p 8800 -b 10.1.1.2 -r 9001
```

```
PHONEMANIA: bind on 10.1.1.1:8800
PHONEMANIA: connect on 10.1.1.2:9001
..
```

- When a client requests a quotation at PhoneMania's entry point (port 8800), PhoneMania actually uses RingBell's back-end server to answer (port 9001). Simply put, PhoneMania gets paid by using RingBell's resources:

```
[colby demo]$ ./TelecomClient -h 10.1.1.1
-p 8800
Connecting to : 10.1.1.1:8800

Requesting quotation for Ericsson
Quote Ericsson
..
[munster demo]$
..
RINGBELL backend : processing quotation request for
                   Ericsson
RINGBELL backend : quotation for Ericsson is 83
Quote Ericsson
```

To prevent this, we propose to subdivide the shared cluster securely into disjoint zones with DSI. Next, we show step by step how to use DSI to do this.

Installing and Configuring DSI

First, we need to install DSI on all nodes of the cluster. After downloading the latest DSI tarball from SourceForge (see Resources), DSI should compile on your machine, as it uses the standard configure and make strategy. We detail how to build and install DSI in the DSI documentation found on the SourceForge site.

You should run the Security Manager on each node. For our two-node cluster, this means it runs on colby and munster:

```
[colby]$ cd ~/dsi
[colby]$ source dsi_setup.sh
[colby]$ ~/dsi/bin/dsiSecManager
```

To simplify, colby also acts as a security server:

```
[colby]$ cd ~/dsi
[colby]$ source dsi_setup.sh
[colby]$ ~/dsi/bin/dsiSecServer
```

The SS and SMs communicate with each other using CORBA event channels.

We load the DSI kernel module DSM on each node to enforce security at the kernel level:

```
$ cd ~/dsi/lsm
$ su root
Password:
# ./load
```

```
# /sbin/lsmod
Module                  Size  Used by    Not tainted
dsm                    36332   0 (unused)
...
```

Then, we configure DSI by defining different IP addresses used on each node for secure and nonsecure communications. To do so, we wrote a tool called DciInit; see the DSI documentation at the SourceForge site for more details on the format of the dci_policy.conf file and how to use DciInit:

```
$ cd ~/dsi/user/tools
$ ./DciInit ~/dsi/etc/dci_policy.conf
```

The Solution: Setting Up the Virtual Subclusters

Basically, to create disjoint virtual subclusters, you need to assign different ScIDs to PhoneMania's resources (in our example, ScID=10) and RingBell's resources (ScID=20). Then, add new rules to DSP to restrict any connection from the zone defined by ScID 10 to the zone defined by ScID 20 and vice versa. By organizing the resources of each operator in separate groups, without any possible connection between them, we actually achieve a virtual subdivision of the cluster. Additionally, the administrator can create another zone defined by ScID 30 with privileges to access both subclusters for administrative purposes.

First, let's assign the ScIDs of each binary on each node (using the DSI SetSID tool):

```
$ ~/dsi/user/tools/SetSID PhoneManiaEP 10
Changing from SID 0 to SID 10
$ ~/dsi/user/tools/SetSID PhoneManiaBE 10
Changing from SID 0 to SID 10
$ ~/dsi/user/tools/SetSID RingBellEP 20
Changing from SID 0 to SID 20
$ ~/dsi/user/tools/SetSID RingBellBE 20
Changing from SID 0 to SID 20
$ ~/dsi/user/tools/ls_dsi .
PERMISSION      USER    GROUP   BSID  FILE
-rwxr-xr-x      lmcaxpr install 10    PhoneManiaBE
-rwxr-xr-x      lmcaxpr install 20    RingBellBE
-rwxr-xr-x      lmcaxpr install 10    PhoneManiaEP
-rwxr-xr-x      lmcaxpr install 20    RingBellEP
```

When DSM is loaded, it enforces default permissive security rules. To achieve cluster subdivision, we have to edit the DSP file ~/dsi/etc/SampleDSP.xml and replace all the existing security rules with our own.

PhoneMania's sockets are assigned ScID=10, and RingBell uses ScID=20. The following rule assigns ScID=10 to PhoneMania's entry point UDP socket (port 8800):

```
<class_SOCKET_INIT_rule>
  <protocol>UDP</protocol>
  <port>8800</port>
```

```
    <SnID>ALL</SnID>
    <ScID>10</ScID>
  </class_SOCKET_INIT_rule>
```

We need three other similar rules: one for PhoneMania's back-end server and two others for ScID=20 for RingBell.

Then, PhoneMania's processes (source ScID=10) are allowed to create, send or receive messages on sockets they own (that is, with target ScID=10):

```
  <class_SOCKET_rule>
    <sScID>10</sScID>
    <sSnID>ALL</sSnID>
    <tScID>10</tScID>
    <tSnID>ALL</tSnID>
    <allow>CREATE CONNECT LISTEN RECEIVE SEND</allow>
  </class_SOCKET_rule>
```

Create a similar rule for RingBell's processes.

Of course, communication between ScID=10 and 20 must be denied. This is done simply by setting no socket permissions at all between those ScIDs:

```
  <class_SOCKET_rule>
    <sScID>10</sScID>
    <sSnID>ALL</sSnID>
    <tScID>20</tScID>
    <tSnID>ALL</tSnID>
    <allow></allow>
  </class_SOCKET_rule>
```

Create a similar rule between source ScID=20 and target ScID=10.

Back-end and entry point servers of a given operator may be located on different nodes of a cluster; remember, we're sharing a cluster, not dedicating one node to RingBell and another to PhoneMania. Hence, processes of PhoneMania (source ScID=10) must be able to communicate with other PhoneMania processes (target ScID=10) through the network. The same holds true for RingBell:

```
  <class_NETWORK_rule>
    <sScID>10</sScID>
    <sSnID>ALL</sSnID>
    <tScID>10</tScID>
    <tSnID>ALL</tSnID>
    <deny>NETWORK_RECEIVE</deny>
  </class_NETWORK_rule>
```

Finally, PhoneMania (ScID=10) and RingBell (ScID=10) processes usually are launched from a shell (default ScID=2). So, basically, we need to allow the shell to create a new process. This is done with a transition rule:

```
<class_TRANSITION_rule>
  <parent_ScID> 2 </parent_ScID>
  <SnID>ALL</SnID>
  <binary_ScID>10</binary_ScID>
  <new_ScID>10</new_ScID>
</class_TRANSITION_rule>
```

The binary_ScID is the ScID explicitly assigned to the binary. Remember, we assigned ScIDs to PhoneManiaBE or PhoneManiaEP using SetSID. The new_ScID is the ScID assigned to the new process also created. As access to sockets 8800 and 8801 is granted only to ScID=10, for PhoneMania, the new process should be assigned ScID=10. A similar rule for RingBell should be created.

This is all we need in the DSP—12 security rules. Then, we update the security policy in the whole cluster by sending an update event to the security server:

```
[colby]$ cd ~/dsi/SS/test/demoSecOM
[colby]$ ./dsiUpdatePolicy ~/dsi/etc/DSP.xml
```

The security server reads the updated DSP file (located in ~/dsi/etc/DSP.xml) and displays warnings if there are syntax errors. Finally, it automatically sends updates to each security manager; no need to log in to each machine to update the security policy manually or develop your own version of system management Perl-based software. This feature can be a benediction when you have a cluster of hundreds of nodes physically spread out around the world (think grid computing).

Now, it's time to try the case again in which PhoneMania forwards requests to RingBell's back-end server:

```
[colby demo]$ ./TelecomClient  -h 10.1.1.1 -p 8800
Requesting quotation for Ericsson
Quote Ericsson
...
[colby demo]$ ./PhoneManiaEP -h 10.1.1.1 -p 8800
-b 10.1.1.2 -r 9001
PHONEMANIA: bind/connect on 10.1.1.1:8800 = 0
PHONEMANIA: bind/connect on 10.1.1.2:9001 = 0
Quote Ericsson

Quotation request received
...
[munster demo]$ ./RingBellBE -h 10.1.1.2 -p 9001
RINGBELL: bind on 10.1.1.2:9001
...
```

On the other node (munster), we notice that RingBell's back-end server is no longer handling PhoneMania's requests, although PhoneMania illicitly redirects them to RingBell. You can use logs generated by DSI, located in /var/log/messages to trace illicit requests:

```
May  6 07:47:31 munster kernel: DSI-LSM MODULE -
dsi_sock_rcv_skb check permission sscid 10
ssnid 1 tscid 20
May  6 07:47:31 munster kernel: DSI-LSM MODULE
Error - dsi_sock_rcv_skb - No Permission
```

Conclusion

We have shown a practical solution for sharing a cluster securely among different applications belonging to different users. The DSI Project allows users to create disjoint security zones for each application in the cluster easily. Once DSI is installed on the cluster, the effort needed to create a new security zone for new applications scales down to setting appropriate ScIDs to binaries and including corresponding rules in the DSP file. Source code modification is not required and probably would be impossible anyway.

Resources for this article: /article/7688.

Makan Pourzandi (makan.pourzandi@ericsson.ca) works for Ericsson Research Canada in the Open Systems Research Department. His research domains are security, cluster computing and component-based methods for distributed programming. He received his Doctoral degree in Parallel Computing in 1995 from the University of Lyon, France.

Axelle Apvrille (axelle.apvrille@ericsson.ca) currently works for Ericsson Research Canada in the Open Systems Research Department. Her research interests are cryptography, security protocols and distributed security. She received her Computer Science Engineering degree in 1996 at ENSEIRB, Bordeaux, France.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Introduction to Sound Programming with ALSA

Jeff Tranter

Issue #126, October 2004

Make maximum use of all the functionality in the new 2.6 kernel sound architecture using a simple API.

ALSA stands for the Advanced Linux Sound Architecture. It consists of a set of kernel drivers, an application programming interface (API) library and utility programs for supporting sound under Linux. In this article, I present a brief overview of the ALSA Project and its software components. The focus is on programming the PCM interfaces of ALSA, including programming examples with which you can experiment.

You may want to explore ALSA simply because it is new, but it is not the only sound API available. ALSA is a good choice if you are performing low-level audio functions for maximum control and performance or want to make use of special features not supported by other sound APIs. If you already have written an audio application, you may want to add native support for the ALSA sound drivers. If your primary interest isn't audio and you simply want to play sound files, using one of the higher-level sound toolkits, such as SDL, OpenAL or those provided in desktop environments, may be a better choice. By using ALSA you are restricted to using systems running a Linux kernel with ALSA support.

History of ALSA

The ALSA Project was started because the sound drivers in the Linux kernel (OSS/Free drivers) were not being maintained actively and were lagging behind the capabilities of new sound technology. Jaroslav Kysela, who previously had written a sound card driver, started the project. Over time, more developers joined, support for many sound cards was added and the structure of the API was refined.

During development of the 2.5 series of Linux kernel, ALSA was merged into the official kernel source. With the release of the 2.6 kernel, ALSA will be part of the stable Linux kernel and should be in wide use.

Digital Audio Basics

Sound, consisting of waves of varying air pressure, is converted to its electrical form by a transducer, such as a microphone. An analog-to-digital converter (ADC) converts the analog voltages into discrete values, called samples, at regular intervals in time, known as the sampling rate. By sending the samples to a digital-to-analog converter and an output transducer, such as a loudspeaker, the original sound can be reproduced.

The size of the samples, expressed in bits, is one factor that determines how accurately the sound is represented in digital form. The other major factor affecting sound quality is the sampling rate. The Nyquist Theorem states that the highest frequency that can be represented accurately is at most one-half the sampling rate.

ALSA Basics

ALSA consists of a series of kernel device drivers for many different sound cards, and it also provides an API library, libasound. Application developers are encouraged to program using the library API and not the kernel interface. The library provides a higher-level and more developer-friendly programming interface along with a logical naming of devices so that developers do not need to be aware of low-level details such as device files.

In contrast, OSS/Free drivers are programmed at the kernel system call level and require the developer to specify device filenames and perform many functions using ioctl calls. For backward compatibility, ALSA provides kernel modules that emulate the OSS/Free sound drivers, so most existing sound applications continue to run unchanged. An emulation wrapper library, libaoss, is available to emulate the OSS/Free API without kernel modules.

ALSA has a capability called plugins that allows extension to new devices, including virtual devices implemented entirely in software. ALSA provides a number of command-line utilities, including a mixer, sound file player and tools for controlling special features of specific sound cards.

ALSA Architecture

The ALSA API can be broken down into the major interfaces it supports:

- Control interface: a general-purpose facility for managing registers of sound cards and querying the available devices.
- PCM interface: the interface for managing digital audio capture and playback. The rest of this article focuses on this interface, as it is the one most commonly used for digital audio applications.
- Raw MIDI interface: supports MIDI (Musical Instrument Digital Interface), a standard for electronic musical instruments. This API provides access to a MIDI bus on a sound card. The raw interface works directly with the MIDI events, and the programmer is responsible for managing the protocol and timing.
- Timer interface: provides access to timing hardware on sound cards used for synchronizing sound events.
- Sequencer interface: a higher-level interface for MIDI programming and sound synthesis than the raw MIDI interface. It handles much of the MIDI protocol and timing.
- Mixer interface: controls the devices on sound cards that route signals and control volume levels. It is built on top of the control interface.

Device Naming

The library API works with logical device names rather than device files. The device names can be real hardware devices or plugins. Hardware devices use the format hw:*i*,*j*, where *i* is the card number and *j* is the device on that card. The first sound device is hw:0,0. The alias default refers to the first sound device and is used in all of the examples in this article. Plugins use other unique names; plughw:, for example, is a plugin that provides access to the hardware device but provides features, such as sampling rate conversion, in software for hardware that does not directly support it. The dmix and dshare plugins allow you to downmix several streams and split a single stream dynamically among different applications.

Sound Buffers and Data Transfer

A sound card has a hardware buffer that stores recorded samples. When the buffer is sufficiently full, it generates an interrupt. The kernel sound driver then uses direct memory access (DMA) to transfer samples to an application buffer in memory. Similarly, for playback, another application buffer is transferred from memory to the sound card's hardware buffer using DMA.

These hardware buffers are ring buffers, meaning the data wraps back to the start when the end of the buffer is reached. A pointer is maintained to keep track of the current positions in both the hardware buffer and the application buffer. Outside of the kernel, only the application buffer is of interest, so from here on we discuss only the application buffer.

The size of the buffer can be programmed by ALSA library calls. The buffer can be quite large, and transferring it in one operation could result in unacceptable delays, called latency. To solve this, ALSA splits the buffer up into a series of periods (called fragments in OSS/Free) and transfers the data in units of a period.

A period stores frames, each of which contains the samples captured at one point in time. For a stereo device, the frame would contain samples for two channels. Figure 1 illustrates the breakdown of a buffer into periods, frames and samples with some hypothetical values. Here, left and right channel information is stored alternately within a frame; this is called interleaved mode. A non-interleaved mode, where all the sample data for one channel is stored followed by the data for the next channel, also is supported.

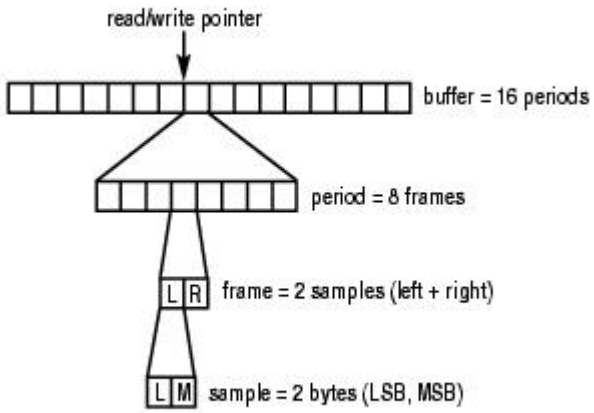


Figure 1. The Application Buffer

Over and Under Run

When a sound device is active, data is transferred continuously between the hardware and application buffers. In the case of data capture (recording), if the application does not read the data in the buffer rapidly enough, the circular buffer is overwritten with new data. The resulting data loss is known as overrun. During playback, if the application does not pass data into the buffer quickly enough, it becomes starved for data, resulting in an error called underrun. The ALSA documentation sometimes refers to both of these conditions using the term XRUN. Properly designed applications can minimize XRUN and recover if it occurs.

A Typical Sound Application

Programs that use the PCM interface generally follow this pseudo-code:

```
open interface for capture or playback
set hardware parameters
(access mode, data format, channels, rate, etc.)
while there is data to be processed:
   read PCM data (capture)
   or write PCM data (playback)
close interface
```

We look at some working code in the following sections. I recommend you compile and run these on your Linux system, look at the output and try some of the suggested modifications. The full listings for the example programs that accompany this article are available for download from ftp.linuxjournal.com/pub/lj/listings/issue126/6735.tgz.

Listing 1. Display Some PCM Types and Formats

```
#include <alsa/asoundlib.h>

int main() {
  int val;

  printf("ALSA library version: %s\n",
         SND_LIB_VERSION_STR);

  printf("\nPCM stream types:\n");
  for (val = 0; val <= SND_PCM_STREAM_LAST; val++)
    printf("  %s\n",
      snd_pcm_stream_name((snd_pcm_stream_t)val));

  printf("\nPCM access types:\n");
  for (val = 0; val <= SND_PCM_ACCESS_LAST; val++)
    printf("  %s\n",
      snd_pcm_access_name((snd_pcm_access_t)val));

  printf("\nPCM formats:\n");
  for (val = 0; val <= SND_PCM_FORMAT_LAST; val++)
    if (snd_pcm_format_name((snd_pcm_format_t)val)
      != NULL)
      printf("  %s (%s)\n",
        snd_pcm_format_name((snd_pcm_format_t)val),
        snd_pcm_format_description(
                        (snd_pcm_format_t)val));

  printf("\nPCM subformats:\n");
  for (val = 0; val <= SND_PCM_SUBFORMAT_LAST;
       val++)
    printf("  %s (%s)\n",
      snd_pcm_subformat_name((
        snd_pcm_subformat_t)val),
      snd_pcm_subformat_description((
        snd_pcm_subformat_t)val));

  printf("\nPCM states:\n");
  for (val = 0; val <= SND_PCM_STATE_LAST; val++)
    printf("  %s\n",
           snd_pcm_state_name((snd_pcm_state_t)val));

  return 0;
}
```

Listing 1 displays some of the PCM data types and parameters used by ALSA. The first requirement is to include the header file that brings in the definitions for all of the ALSA library functions. One of the definitions is the version of ALSA, which is displayed.

The remainder of the program iterates through a number of PCM data types, starting with the stream types. ALSA provides symbolic names for the last enumerated value and a utility function that returns a descriptive string for a value. As you can see in the output, ALSA supports many different data formats, 38 for the version of ALSA on my system.

The program must be linked with the ALSA library, libasound, to run. Typically, you would add the option -lasound on the linker command line. Some ALSA library functions use the dlopen function and floating-point operations, so you also may need to add -ldl and -lm.

Listing 2. Opening PCM Device and Setting Parameters

```
/*

This example opens the default PCM device, sets
some parameters, and then displays the value
of most of the hardware parameters. It does not
perform any sound playback or recording.

*/

/* Use the newer ALSA API */
#define ALSA_PCM_NEW_HW_PARAMS_API

/* All of the ALSA library API is defined
 * in this header */
#include <alsa/asoundlib.h>

int main() {
  int rc;
  snd_pcm_t *handle;
  snd_pcm_hw_params_t *params;
  unsigned int val, val2;
  int dir;
  snd_pcm_uframes_t frames;

  /* Open PCM device for playback. */
  rc = snd_pcm_open(&handle, "default",
                    SND_PCM_STREAM_PLAYBACK, 0);
  if (rc < 0) {
    fprintf(stderr,
            "unable to open pcm device: %s\n",
            snd_strerror(rc));
    exit(1);
  }

  /* Allocate a hardware parameters object. */
  snd_pcm_hw_params_alloca(&params);

  /* Fill it in with default values. */
  snd_pcm_hw_params_any(handle, params);

  /* Set the desired hardware parameters. */

  /* Interleaved mode */
  snd_pcm_hw_params_set_access(handle, params,
                      SND_PCM_ACCESS_RW_INTERLEAVED);
```

```
    /* Signed 16-bit little-endian format */
    snd_pcm_hw_params_set_format(handle, params,
                                SND_PCM_FORMAT_S16_LE);

    /* Two channels (stereo) */
    snd_pcm_hw_params_set_channels(handle, params, 2);

    /* 44100 bits/second sampling rate (CD quality) */
    val = 44100;
    snd_pcm_hw_params_set_rate_near(handle,
                                    params, &val, &dir);

    /* Write the parameters to the driver */
    rc = snd_pcm_hw_params(handle, params);
    if (rc < 0) {
      fprintf(stderr,
              "unable to set hw parameters: %s\n",
              snd_strerror(rc));
      exit(1);
    }

    /* Display information about the PCM interface */

    printf("PCM handle name = '%s'\n",
           snd_pcm_name(handle));

    printf("PCM state = %s\n",
           snd_pcm_state_name(snd_pcm_state(handle)));

    snd_pcm_hw_params_get_access(params,
                          (snd_pcm_access_t *) &val);
    printf("access type = %s\n",
           snd_pcm_access_name((snd_pcm_access_t)val));

    snd_pcm_hw_params_get_format(params, &val);
    printf("format = '%s' (%s)\n",
      snd_pcm_format_name((snd_pcm_format_t)val),
      snd_pcm_format_description(
                             (snd_pcm_format_t)val));

    snd_pcm_hw_params_get_subformat(params,
                         (snd_pcm_subformat_t *)&val);
    printf("subformat = '%s' (%s)\n",
      snd_pcm_subformat_name((snd_pcm_subformat_t)val),
      snd_pcm_subformat_description(
                           (snd_pcm_subformat_t)val));

    snd_pcm_hw_params_get_channels(params, &val);
    printf("channels = %d\n", val);

    snd_pcm_hw_params_get_rate(params, &val, &dir);
    printf("rate = %d bps\n", val);

    snd_pcm_hw_params_get_period_time(params,
                                      &val, &dir);
    printf("period time = %d us\n", val);

    snd_pcm_hw_params_get_period_size(params,
                                      &frames, &dir);
    printf("period size = %d frames\n", (int)frames);

    snd_pcm_hw_params_get_buffer_time(params,
                                      &val, &dir);
    printf("buffer time = %d us\n", val);

    snd_pcm_hw_params_get_buffer_size(params,
                          (snd_pcm_uframes_t *) &val);
    printf("buffer size = %d frames\n", val);

    snd_pcm_hw_params_get_periods(params, &val, &dir);
    printf("periods per buffer = %d frames\n", val);

    snd_pcm_hw_params_get_rate_numden(params,
                                      &val, &val2);
    printf("exact rate = %d/%d bps\n", val, val2);

    val = snd_pcm_hw_params_get_sbits(params);
    printf("significant bits = %d\n", val);
```

```
    snd_pcm_hw_params_get_tick_time(params,
                                    &val, &dir);
    printf("tick time = %d us\n", val);

    val = snd_pcm_hw_params_is_batch(params);
    printf("is batch = %d\n", val);

    val = snd_pcm_hw_params_is_block_transfer(params);
    printf("is block transfer = %d\n", val);

    val = snd_pcm_hw_params_is_double(params);
    printf("is double = %d\n", val);

    val = snd_pcm_hw_params_is_half_duplex(params);
    printf("is half duplex = %d\n", val);

    val = snd_pcm_hw_params_is_joint_duplex(params);
    printf("is joint duplex = %d\n", val);

    val = snd_pcm_hw_params_can_overrange(params);
    printf("can overrange = %d\n", val);

    val = snd_pcm_hw_params_can_mmap_sample_resolution(params);
    printf("can mmap = %d\n", val);

    val = snd_pcm_hw_params_can_pause(params);
    printf("can pause = %d\n", val);

    val = snd_pcm_hw_params_can_resume(params);
    printf("can resume = %d\n", val);

    val = snd_pcm_hw_params_can_sync_start(params);
    printf("can sync start = %d\n", val);

    snd_pcm_close(handle);

    return 0;
  }
```

Listing 2 opens the default PCM device, sets some parameters and then displays the values of most of the hardware parameters. It does not perform any sound playback or recording. The call to snd_pcm_open opens the default PCM device and sets the access mode to PLAYBACK. This function returns a handle in the first function argument that is used in subsequent calls to manipulate the PCM stream. Like most ALSA library calls, the function returns an integer return status, a negative value indicating an error condition. In this case, we check the return code; if it indicates failure, we display the error message using the snd_strerror function and exit. In the interest of clarity, I have omitted most of the error checking from the example programs. In a production application, one should check the return code of every API call and provide appropriate error handling.

In order to set the hardware parameters for the stream, we need to allocate a variable of type snd_pcm_hw_params_t. We do this with the macro snd_pcm_hw_params_alloca. Next, we initialize the variable using the function snd_pcm_hw_params_any, passing the previously opened PCM stream.

We now set the desired hardware parameters using API calls that take the PCM stream handle, the hardware parameters structure and the parameter value.

We set the stream to interleaved mode, 16-bit sample size, 2 channels and a 44,100 bps sampling rate. In the case of the sampling rate, sound hardware is not always able to support every sampling rate exactly. We use the function snd_pcm_hw_params_set_rate_near to request the nearest supported sampling rate to the requested value. The hardware parameters are not actually made active until we call the function snd_pcm_hw_params.

The rest of the program obtains and displays a number of the PCM stream parameters, including the period and buffer sizes. The results displayed vary somewhat depending on the sound hardware.

After running the program on your system, experiment and make some changes. Change the device name from default to hw:0,0 or plughw: and see whether the results change. Change the hardware parameter values and observe how the displayed results change.

Listing 3. Simple Sound Playback

```
/*

This example reads standard from input and writes
to the default PCM device for 5 seconds of data.

*/

/* Use the newer ALSA API */
#define ALSA_PCM_NEW_HW_PARAMS_API

#include <alsa/asoundlib.h>

int main() {
  long loops;
  int rc;
  int size;
  snd_pcm_t *handle;
  snd_pcm_hw_params_t *params;
  unsigned int val;
  int dir;
  snd_pcm_uframes_t frames;
  char *buffer;

  /* Open PCM device for playback. */
  rc = snd_pcm_open(&handle, "default",
                    SND_PCM_STREAM_PLAYBACK, 0);
  if (rc < 0) {
    fprintf(stderr,
            "unable to open pcm device: %s\n",
            snd_strerror(rc));
    exit(1);
  }

  /* Allocate a hardware parameters object. */
  snd_pcm_hw_params_alloca(&params);

  /* Fill it in with default values. */
  snd_pcm_hw_params_any(handle, params);

  /* Set the desired hardware parameters. */

  /* Interleaved mode */
  snd_pcm_hw_params_set_access(handle, params,
                      SND_PCM_ACCESS_RW_INTERLEAVED);
```

```
    /* Signed 16-bit little-endian format */
    snd_pcm_hw_params_set_format(handle, params,
                                 SND_PCM_FORMAT_S16_LE);

    /* Two channels (stereo) */
    snd_pcm_hw_params_set_channels(handle, params, 2);

    /* 44100 bits/second sampling rate (CD quality) */
    val = 44100;
    snd_pcm_hw_params_set_rate_near(handle, params,
                                    &val, &dir);

    /* Set period size to 32 frames. */
    frames = 32;
    snd_pcm_hw_params_set_period_size_near(handle,
                                 params, &frames, &dir);

    /* Write the parameters to the driver */
    rc = snd_pcm_hw_params(handle, params);
    if (rc < 0) {
      fprintf(stderr,
              "unable to set hw parameters: %s\n",
              snd_strerror(rc));
      exit(1);
    }

    /* Use a buffer large enough to hold one period */
    snd_pcm_hw_params_get_period_size(params, &frames,
                                      &dir);
    size = frames * 4; /* 2 bytes/sample, 2 channels */
    buffer = (char *) malloc(size);

    /* We want to loop for 5 seconds */
    snd_pcm_hw_params_get_period_time(params,
                                      &val, &dir);
    /* 5 seconds in microseconds divided by
     * period time */
    loops = 5000000 / val;

    while (loops > 0) {
      loops--;
      rc = read(0, buffer, size);
      if (rc == 0) {
        fprintf(stderr, "end of file on input\n");
        break;
      } else if (rc != size) {
        fprintf(stderr,
                "short read: read %d bytes\n", rc);
      }
      rc = snd_pcm_writei(handle, buffer, frames);
      if (rc == -EPIPE) {
        /* EPIPE means underrun */
        fprintf(stderr, "underrun occurred\n");
        snd_pcm_prepare(handle);
      } else if (rc < 0) {
        fprintf(stderr,
                "error from writei: %s\n",
                snd_strerror(rc));
      }  else if (rc != (int)frames) {
        fprintf(stderr,
                "short write, write %d frames\n", rc);
      }
    }

    snd_pcm_drain(handle);
    snd_pcm_close(handle);
    free(buffer);

    return 0;
  }
```

Listing 3 extends the previous example by writing sound samples to the sound card to produce playback. In this case we read bytes from standard input,

enough for one period, and write them to the sound card until five seconds of data has been transferred.

The beginning of the program is the same as in the previous example—the PCM device is opened and the hardware parameters are set. We use the period size chosen by ALSA and make this the size of our buffer for storing samples. We then find out that period time so we can calculate how many periods the program should process in order to run for five seconds.

In the loop that manages data, we read from standard input and fill our buffer with one period of samples. We check for and handle errors resulting from reaching the end of file or reading a different number of bytes from what was expected.

To send data to the PCM device, we use the snd_pcm_writei call. It operates much like the kernel write system call, except that the size is specified in frames. We check the return code for a number of error conditions. A return code of EPIPE indicates that underrun occurred, which causes the PCM stream to go into the XRUN state and stop processing data. The standard method to recover from this state is to use the snd_pcm_prepare function call to put the stream in the PREPARED state so it can start again the next time we write data to the stream. If we receive a different error result, we display the error code and continue. Finally, if the number of frames written is not what was expected, we display an error message.

The program loops until five seconds' worth of frames has been transferred or end of file read occurs on the input. We then call snd_pcm_drain to allow any pending sound samples to be transferred, then close the stream. We free the dynamically allocated buffer and exit.

We should see that the program is not useful unless the input is redirected to something other than a console. Try running it with the device /dev/urandom, which produces random data, like this:

```
./example3 < /dev/urandom
```

The random data should produce white noise for five seconds.

Next, try redirecting the input to /dev/null or /dev/zero and compare the results. Change some parameters, such as the sampling rate and data format, and see how it affects the results.

Listing 4. Simple Sound Recording

```
/*

This example reads from the default PCM device
and writes to standard output for 5 seconds of data.

*/

/* Use the newer ALSA API */
#define ALSA_PCM_NEW_HW_PARAMS_API

#include <alsa/asoundlib.h>

int main() {
  long loops;
  int rc;
  int size;
  snd_pcm_t *handle;
  snd_pcm_hw_params_t *params;
  unsigned int val;
  int dir;
  snd_pcm_uframes_t frames;
  char *buffer;

  /* Open PCM device for recording (capture). */
  rc = snd_pcm_open(&handle, "default",
                    SND_PCM_STREAM_CAPTURE, 0);
  if (rc < 0) {
    fprintf(stderr,
            "unable to open pcm device: %s\n",
            snd_strerror(rc));
    exit(1);
  }

  /* Allocate a hardware parameters object. */
  snd_pcm_hw_params_alloca(&params);

  /* Fill it in with default values. */
  snd_pcm_hw_params_any(handle, params);

  /* Set the desired hardware parameters. */

  /* Interleaved mode */
  snd_pcm_hw_params_set_access(handle, params,
                      SND_PCM_ACCESS_RW_INTERLEAVED);

  /* Signed 16-bit little-endian format */
  snd_pcm_hw_params_set_format(handle, params,
                              SND_PCM_FORMAT_S16_LE);

  /* Two channels (stereo) */
  snd_pcm_hw_params_set_channels(handle, params, 2);

  /* 44100 bits/second sampling rate (CD quality) */
  val = 44100;
  snd_pcm_hw_params_set_rate_near(handle, params,
                                  &val, &dir);

  /* Set period size to 32 frames. */
  frames = 32;
  snd_pcm_hw_params_set_period_size_near(handle,
                              params, &frames, &dir);

  /* Write the parameters to the driver */
  rc = snd_pcm_hw_params(handle, params);
  if (rc < 0) {
    fprintf(stderr,
            "unable to set hw parameters: %s\n",
            snd_strerror(rc));
    exit(1);
  }

  /* Use a buffer large enough to hold one period */
  snd_pcm_hw_params_get_period_size(params,
                                    &frames, &dir);
  size = frames * 4; /* 2 bytes/sample, 2 channels */
```

```
    buffer = (char *) malloc(size);

    /* We want to loop for 5 seconds */
    snd_pcm_hw_params_get_period_time(params,
                                          &val, &dir);
    loops = 5000000 / val;

    while (loops > 0) {
      loops--;
      rc = snd_pcm_readi(handle, buffer, frames);
      if (rc == -EPIPE) {
        /* EPIPE means overrun */
        fprintf(stderr, "overrun occurred\n");
        snd_pcm_prepare(handle);
      } else if (rc < 0) {
        fprintf(stderr,
                "error from read: %s\n",
                snd_strerror(rc));
      } else if (rc != (int)frames) {
        fprintf(stderr, "short read, read %d frames\n", rc);
      }
      rc = write(1, buffer, size);
      if (rc != size)
        fprintf(stderr,
                "short write: wrote %d bytes\n", rc);
    }

    snd_pcm_drain(handle);
    snd_pcm_close(handle);
    free(buffer);

    return 0;
  }
```

Listing 4 is much like Listing 3, except that we perform PCM capture (recording). When we open the PCM stream, we specify the mode as SND_PCM_STREAM_CAPTURE. In the main processing loop, we read the samples from the sound hardware using snd_pcm_readi and write it to standard output using write. We check for overrun and handle it in the same manner as we did underrun in Listing 3.

Running Listing 4 records approximately five seconds of data and sends it to standard out; you should redirect it to a file. If you have a microphone connected to your sound card, use a mixer program to set the recording source and level. Alternatively, you can run a CD player program and set the recording source to CD. Try running Listing 4 and redirecting the output to a file. You then can run Listing 3 to play back the data:

```
./listing4 > sound.raw
./listing3 < sound.raw
```

If your sound card supports full duplex sound, you should be able to pipe the programs together and hear the recorded sound coming out of the sound card by typing: `./listing4 | ./listing3`. By changing the PCM parameters you can experiment with the effect of sampling rates and formats.

Advanced Features

In the previous examples, the PCM streams were operating in blocking mode, that is, the calls would not return until the data had been transferred. In an interactive event-driven application, this situation could lock up the application for unacceptably long periods of time. ALSA allows opening a stream in nonblocking mode where the read and write functions return immediately. If data transfers are pending and the calls cannot be processed, ALSA returns an error code of EBUSY.

Many graphical applications use callbacks to handle events. ALSA supports opening a PCM stream in asynchronous mode. This allows registering a callback function to be called when a period of sample data has been transferred.

The snd_pcm_readi and snd_pcm_writei calls used here are similar to the Linux read and write system calls. The letter i indicates that the frames are interleaved; corresponding functions exist for non-interleaved mode. Many devices under Linux also support the mmap system call, which maps them into memory where they can be manipulated with pointers. Finally, ALSA supports opening a PCM channel in mmap mode, which allows efficient zero copy access to sound data.

Conclusion

I hope this article has motivated you to try some ALSA programming. As the 2.6 kernel becomes commonly used by Linux distributions, ALSA should become more widely used, and its advanced features should help Linux audio applications move forward.

My thanks to Jaroslav Kysela and Takashi Iwai for reviewing a draft of this article and providing me with useful comments.

Resources for this article: /article/7705.

Jeff Tranter has been using, writing about and contributing to Linux since 1992. He works for Xandros Corporation in Ottawa, Canada.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The Politics of Porting

Stephen C. Forster

Issue #126, October 2004

The flagship application was headed for the rocks. One man seized the wheel and guided the ship to safety—but will he be keelhauled for acting without orders?

At each of the companies where I have worked in the past eight years, I have introduced Linux in some capacity. It often seemed the obvious choice, but what was obvious to me and other colleagues was not obvious to everyone. Some people needed persuading before they would accept what to them was an unconventional solution. In most cases, benchmarks and cost factors were enough to tip the balance. Novelty and openness to new ideas were sufficient for others. But, for a stubborn minority, nothing could persuade them. Although they participated in the debate, it appeared that for them discussion was simply a tactic to avoid implementation.

Faced with such opposition, arguments are pointless and demonstrations are a waste of time. In those situations, other tactics are called for. Here I describe how I persuaded my employer to port the company's flagship application to Linux literally before the opposition in management knew what was happening.

Background

In 1998, I was UNIX and Oracle administrator for a company called Constellar Ltd. They specialised in data migration and Enterprise Application Integration (EAI) solutions for Global 2000 companies. What does that mean? At its simplest, it might mean transforming the data in a legacy mainframe database for insertion into a newly prepared database. Such tasks are by their nature repetitive and time consuming. They are ideal candidates for automation and that is where the Constellar Hub comes in.

Add a few extra offices in different parts of the world, more databases for customer data, warehouse inventories and order tracking and imagine how

these assorted data stores and applications could be made to work together seamlessly. Throw in a live environment and a company that relies on accurate available data, and you begin to understand how complex the task could become.

The main application was a client server called the Constellar Hub, which was designed to extract data from disparate sources, processing, weeding and integrating it on the fly before writing or streaming the transformed data to its destination.

The server relied on an Oracle database to store data in transit, as well as the metadata in which the business rules were held, data types were defined and the relationship between the multifarious data sources and destinations were stored.

The combined transient data and metadata were held in upward of 200 database tables with many triggers and constraints to enforce the integrity of the data as it flowed through the bowels of the application. The engine of the application consisted primarily of C and SQL code, but an API provided the opportunity to extend functionality if required.

On a large project, two or many Hub servers might be required, sometimes located at remote sites around the world, feeding or being fed data through frame relay or less likely the Internet. Typically, an average project would process many gigabytes of data and usually hundreds, thousands or, more rarely, tens of thousands of business rules would be defined.

In many cases, the whole project of which the Hub was only a part—albeit a central part—would cost in the millions and involve dozens of technical and business staff.

Prospective customers would be given a relatively simple demonstration of the tool running on test data, typically on Sun hardware. The server had been ported to various other shades of UNIX, including Dynix, Data General UNIX, AIX and Digital UNIX.

Windows NT Makes Its Entrance

It was also around that time that Windows NT was being promoted as an enterprise-ready OS destined to take over in many areas previously thought to be the exclusive domain of UNIX. Indeed, our vice president of engineering was convinced that the future was Microsoft and told us so. In the future, he said, most if not all our sales would be generated from the Windows NT version of the Hub. As a first move, they already were working on a Windows NT port and had announced the beta version in February 1998. Such a view was

understandable, because the cost of a Windows NT box versus the various UNIX platforms tended to highlight only the expense of UNIX solutions. Linux was destined to change all that.

By that time, I already was using Linux on all my home PCs as well as my workplace workstation. I also followed Linux developments fairly closely, especially any that might affect the adoption of Linux in the enterprise workplace. Like many other Linux advocates, I saw the possibility of a computing future that might not be restricted to the products of a single company imposed on a captive customer base and wondered if the Constellar Hub could be ported to Linux.

There were two problems, however. First, the Hub made extensive use of Oracle databases and Oracle-specific features, and it would be a nightmare porting those dependencies to a Linux-friendly equivalent. Second, and this was a more difficult hurdle to clear, a few people in the company were not predisposed to see the opportunities presented by a Linux port.

I vaguely had suggested to one or two members of middle management that perhaps there ought to be a Linux port, but the response I received in return was along the lines that Linux is an amazing product for a bunch of amateurs to have produced, but it wasn't quite ready for the big leagues, and the company couldn't afford the time and expense of porting to it when there was serious business to deal with. I have heard that argument many times since then and have learned it is largely an excuse for not thinking.

Two things changed the nature of the ball game. First, in September 1998, Larry Ellison announced, with excellent timing, that Oracle would release a Linux version of its database server and made it free for download for development use. This had a dual effect. It demonstrated that some very large companies were taking Linux seriously, and at the same time, it made the possibility of a Linux port of the Constellar Hub feasible.

I decided that if I couldn't persuade them to port the Hub I would do it myself, so I copied the Hub source code and set about providing a proof of concept. I figured this either would get me the sack or provide a fait accompli with which those in opposition could not argue.

I "Steal" the Code

At this point I should say "Don't do what I did." It is illegal to copy your employer's source code without permission, and you are likely to find yourself at the wrong end of a lawsuit if it all goes belly up, but sometimes dire circumstances require drastic solutions.

In late November 1998, I checked out a full set of source code and, along with a newly downloaded copy of Oracle 8.0.4, I set up the development environment on my PC at home; using the unedited Makefile from the Solaris development directory, I ran `make`. Not surprisingly, it all fell in a heap, throwing up pages of compile errors.

I devoted an hour or two most evenings to stepping through the errors and debugging them. Those I didn't understand I presented to Richard Glover, one of the senior UNIX developers at Constellar. He also was the release manager for the various UNIX versions of the Hub, so he was a good person to know.

Most of the problems revolved around the Oracle ProC precompiler, which didn't recognise some Linux-specific directives, such as the include_next statements in the Linux header files. This was resolved by copying the relevant files, stripping the `#include_next` statements out and placing the location of our customised version of these files first in the include path.

There also were assorted issues with library paths. These were trivial and related solely to the setup on my home PC. They were solved by adding entries to the LD_LIBRARY_PATH shell variable exactly as it would be with Solaris. Some macro definitions—ULONG_MAX, INT_MIN, INT_MAX, LONG_MAX—were missing from the copy of Red Hat that I was using at the time. I copied them from the relevant Solaris headers as a temporary fix.

I left some minor problems unresolved because my resources were somewhat limited. For instance, the Hub was required to connect to many and varied sources of data, and some of these required the use of IBM MQ series libraries, which I did not have available at the time. In any case, I did not have the facilities to test such functionality at home. I edited the main Makefile to disable the linking of these libraries and accepted that my version of the Hub would be a lightweight, more dynamic kind of Hub. If it compiled at all, I would be happy.

Some differences found in the Linux version of various utilities were sorted out by editing scripts to reflect the proper invocation of particular utilities. I worked my way around some of them, and I ignored others to save time. Richard would later track down the source of all these problems and make allowances for them in the platform-specific sections of various build scripts in the official build environment at work. Those utilities that required changes to the standard build scripts included various compiler and linker flags. We were obliged to use the Sun compiler for the official Solaris build, so moving to Linux and GCC required a bit of translation with regard to the flags and switches. The use of df, the shell built-in echo command, ftp, ldd, mknod, nm, ps and lex/yacc (used for the proprietary Transformation Definition Language) all required changes when invoked.

As Richard pointed out a few weeks later, the set -o posix option in bash resolved nearly all the shell script differences, and in any case, the command-line differences were quite trivial and the solution obvious. In some cases, it was simply a case of providing Berkeley versions of the utilities rather than the System V ones, or vice versa. The situation would be different today, because many of the GNU utilities standard on Linux are now included as standard with Solaris, so porting between the two has become even simpler.

In all, it probably took about two man-days to get a rough-and-ready port with a binary that at least looked like an executable.

The Smoke Test

I had a binary, now I needed to test it. I checked out a copy of the suite of test scripts and data used in the daily smoke test at work. After setting the environment variables for the smoke test, I set the test harness running.

The usual smoke test consisted of more than 200 transactions, and on the other platforms would take anything from six hours for UNIX versions to more than 24 hours for the Windows NT version—this is not good when testing a daily build. I ran the test on my home-built Linux port, and within a minute or two I had errors scrolling up the screen. Disaster. I checked the error logs, but they were no help. For some reason, the logs didn't provide any diagnostics for the first few dozen transactions, and this confused me.

It took a while to figure out that the reason there were no diagnostics in the error log for the initial set of transactions was simply because the Linux Hub already had run the first few dozen transactions successfully! This initially was hard to believe, because the same set of transactions took considerably longer to run on most of the other platforms to which the Hub was ported. This meant that my rough homebrew Linux version of the Hub was, without tuning, already faster on my home PC than some of the other UNIX ports, and several times faster than the Windows NT port. Even allowing for the differences between the hardware specifications on which these varied ports ran, it looked good.

I felt sure that if these results could be reproduced in the workplace, there would have to be an official Linux port.

Politics

My suggestion that the Hub be ported to Linux previously had been met with a lukewarm response from the middle management with whom I had broached the subject, so I decided to adopt a different tactic this time around.

Several others had expressed an interest in finding out more about Linux, so a couple of months previously we had formed a company Linux User Group with a mail list on the company mail server and a meeting to talk tech once a week. I let the other members—Ngozi Mayo, Anthony Durity and Olusola Ojeje—know what I had been up to and they provided useful feedback and advice.

After some thought, I realised that I had no idea who, if anyone, in senior management would lend support to the idea of a Linux port. I decided to e-mail all of them and the members of the board and hope that someone among them would take an interest.

I planned to announce the alpha port in the style of a press release and see if it would get me the sack or not. I wrote a draft release with a number of bullet points and submitted it for review to the members of the LUG. I made corrections, simplified the announcement and toned down the style as suggested, and then I sent off the revised release and waited for a response.

On February 26, 1999, I pointed out in my press release that the port had cost the company nothing, that the Linux port ran the smoke test several times faster than the Windows NT version, and at least as fast as the other UNIX ports, that there was an increasing trend toward Linux ports of enterprise software and that now, 1999, was the time to get on board. I requested that the company supply hardware to enable an official company Linux port and that it be defined as a tier-three platform to give it official status. Quotes from IDC, highlighting a 212% increase in Linux sales that year and 17% share of the enterprise market, helped shock the complacent into life. For many of the e-mail recipients these facts and figures were a revelation.

The e-mail generated an immediate response and much discussion. Some people sent congratulations and others speculated about possible marketing angles. One senior Sales Consulting Manager questioned the need for a Linux port at all, pointing out that in the previous two years not one customer had requested such a thing. We would all still be riding round in horse and carriage if that attitude prevailed. Someone suggested that “Enterprise deals are closed on enterprise platforms” without then explaining why Windows NT was an “Enterprise” platform or how Linux was not. Others didn't quite understand what Linux was and the LUG did their best to explain.

The final word went to our CEO Nic Birtles who proved to be both insightful and encouraging. In his pithy e-mail response to the ongoing debate, he simply asked what had to be done to be able to announce a Linux version of the HUB. Debate over.

We got our hardware for a dedicated Linux development machine and the Linux port became an official tier-three platform. Having produced a rough-and-ready port merely to prove a point, the task of producing a robust Linux binary was given to Richard Glover.

Everything went quiet for a while, and then one day in early May 1999, Richard quietly mentioned in passing that the Linux port had a 100% pass rate the previous night and that it had taken only five hours to complete. Good news. How long did the Windows NT version take? Fifteen hours on a good night, although some nights the process had to be killed because there weren't enough hours in a day.

Another in-house press release for management and senior staff was prepared and sent to announce the beta release and the amazing test results. It garnered a wide range of comments both positive and negative. In the latter category, our Director of European Marketing flirted with both sides in her e-mail: “Great job” but then spoiled it by suggesting that we didn't want to be a trailblazer in the Linux “space”, no matter that the Linux port ran faster than any of the other ports and three times faster than the Windows NT version. Fortunately, most reactions were much more positive and the most important one, the CEO's, was both congratulatory and helpful. He suggested that the Linux port could now become a tier-two platform alongside Windows NT and the other UNIXes. Only Solaris and AIX were tier-one.

By July 1999, the Linux port was added to the official application CD.

By the time we had an official Linux release of the Constellar Hub on the CD, 25% of the UK workforce had joined the Linux User Group, and the Linux objectors were a small and subdued minority keeping their thoughts to themselves.

The Constellar Hub subsequently was bought by Data Mirror who continues to offer and support the Linux version, which currently is at version 3.8. Thanks to a bit of guerilla porting and a CEO who knew a bargain when he saw one, this Linux application finally found its way out into the open.

Postscript: Cats and Dogs

So why is it so difficult to convince some people in the face of such revealing benchmarks? The cost of the port was close to zero, the performance was superior to almost all the other ports and the resulting Linux Hub was rock solid.

It think the answer to that question hinges around the personality of the individuals concerned. There seems to be two types of people involved in this

debate, and although each may be amenable to argument, their instinct leaves them predisposed to favour particular forms of software solutions.

One group prefers a clear and rigid hierarchy. They want to know who they are answerable to and who is answerable to them. They want one organisation providing one solution and would like to dispense with the alternatives. They respect the dominant player, whoever it is. They feel at ease with conformity and would prefer it if you would conform too.

On the other hand, the second group values freedom of choice. They tend not to take anything for granted, want to see under the bonnet when considering a product and always are ready to look at alternatives. They certainly don't want to be told what to do but are amenable to argument and debate.

These two groups might be characterised as dogs and cats. The dogs listen to their masters' voice and defend their own territory and that of their master. They know their place and yours and would prefer to keep it that way.

The second group are cats. They don't recognise a master, never mind his voice. There are no limits to their territory, and they range widely. They are curious and seek out new pastures and are at ease with the unfamiliar.

Next time you are having problems introducing Linux into the workplace, the resistance you encounter might have nothing to do with the strengths and weaknesses of the software under discussion. It simply might be that you are a cool cat trying to persuade an old dog to learn new tricks.

Stephen C. Forster has worked in the IT industry for nine years, mostly as a UNIX/Linux administrator, though he writes code and administers Oracle databases when he must. In another life, he also has been in the army and shot news footage of the war in Afghanistan. He can be reached at steve@kidik.net.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Linux Tools for Professional Photography

RW Hawkins

Issue #126, October 2004

Tools are available that allow you to do professional photographic work straight from your Linux desktop.

A new breed of inexpensive, photo-quality products are revolutionizing the photo industry, and Linux is there with the tools to put these products to use. In 1990, while editing my first digital image on a Mac SE, I had visions of one day having all the power of a high-end photo retouching workstation in my own digital studio. Several years later, I had the Mac studio I had dreamed of but still was dependent on an outside service bureau for scanning and printing. Around 1996, I saw an early version of The GIMP and was excited at the possibility of moving my digital studio to Linux. Fast-forward to the present: inexpensive photo scanners and printers are available that offer professional results at affordable prices, and the Linux drivers and tools are there to take advantage. A color management work flow is starting to take shape on Linux, and The GIMP is maturing at a steady pace.

I now can go from scanned transparency through digital retouching to final output, all in my own studio running Linux. This article explains the steps required for creating professional digital prints with Linux by following one of my images through the process. Rather than go into details about compiling and installing Linux packages, I try to give a bird's-eye view of how the different pieces fit together and provide links to more detailed information.

Hardware Requirements

Not much separates the workstation I use for photo editing from a good Linux desktop. Certainly, the more memory you have the better; I recommend 512MB as a minimum, and a fast disk is a real time-saver as well. My photo workstation is a dual-head, 1.7GHz P4 machine with 1GB of RAM and three 36GB, 10K RPM SCSI hard drives. What truly separates a photo editing setup from a generic desktop are the peripherals. Excellent photo-quality scanners and printers are

now available for under $1,000 US. Most of these are supported under Linux, but always check the Linux compatibility pages before you buy one of these specialized devices. When looking for a scanner, cross-check advice from dedicated on-line photo forums, such as Photo.net, with the Linux compatibility information found on the VueScan and SANE Web sites. Because I shoot large format, 4" × 5" film, my choice in desktop scanners is limited to flatbed scanners with transparency adapters. For 35mm film scanning, the dedicated film scanners offer better results than flatbeds. For photo printers, it's hard to beat the current generation of Epson inkjets. The LinuxPrinting.org site provides a great utility to find out whether a particular printer is supported under Linux, and it even suggests the best driver to use.

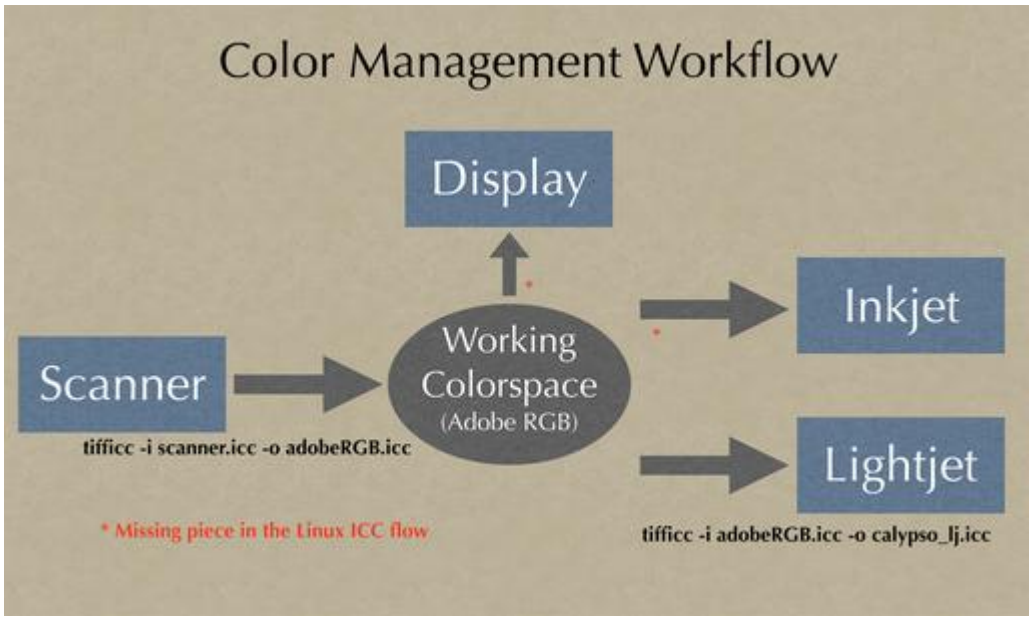


Figure 1. Work-Flow Diagram for Color Management under Linux

Color Communication

Communication is all about having a standard vocabulary that two parties understand, and in the world of color communication the International Color Consortium (ICC) provides this lingua franca. All color devices have their own color space, meaning they are able to reproduce only a certain range of colors. This type of color is known as device-dependent color. In order to convert between different color spaces, a device-independent color space is needed. ICC uses a color description known as the CIE 1931 standard colorimetric observer to describe accurately the color space of any color device. ICC profiles are files that contain the information needed to translate the color space of a particular device to this device-independent color space. A special type of color space, called the working color space, is a device-independent color space used when editing an image. By having one common language and a translator from each different language, complete communication is possible. Although it's helpful to know all about color theory, it is not required to make good color

prints. All you do need to know is that ICC profiles are used to convert the colors from one device to the colors of another device accurately.

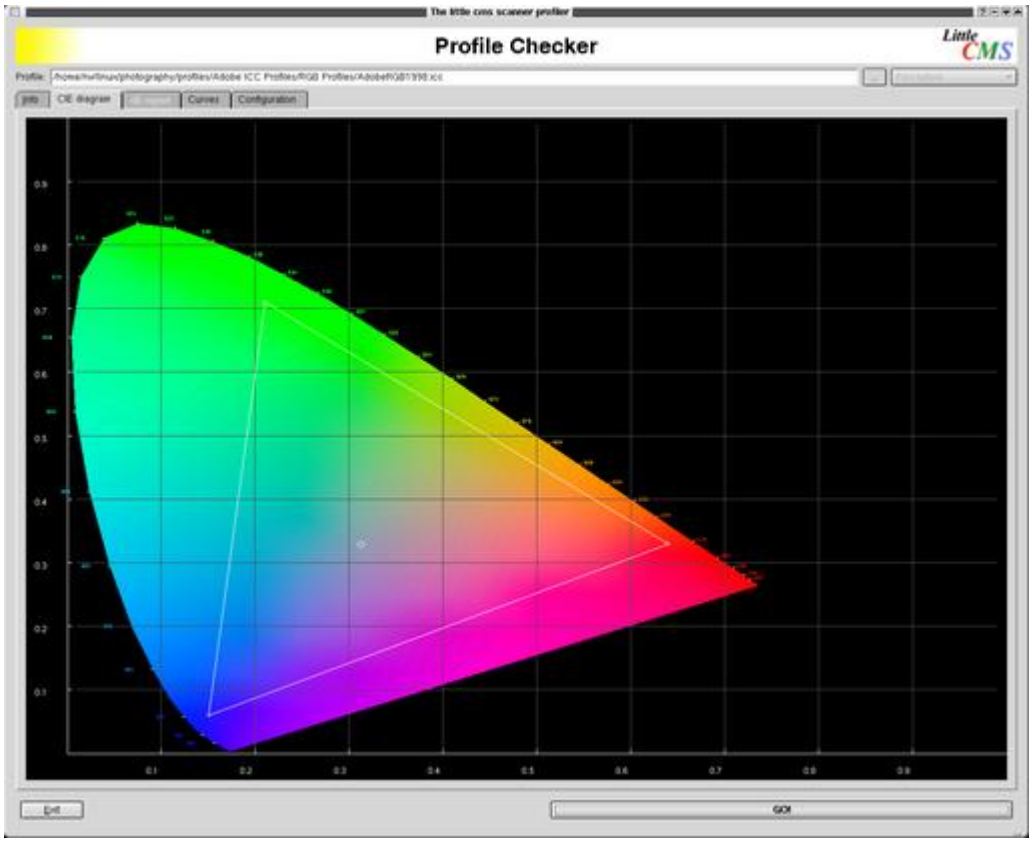


Figure 2. CIE diagram showing visual color space. The triangle shows the color space of the AdobeRGB 1998 working space.

Monitor Calibration

An accurate monitor is vital for working with color photos. Professionals use a device called a colorimeter to measure the actual computer display and create an ICC profile that then is used by the OS to reproduce colors faithfully. Unfortunately, I am aware of no Linux drivers that support any of these devices. Even if you could get a monitor profile, The GIMP does not support using one for monitor display. Certainly, this is one feature in The GIMP I sorely miss. In fact, The GIMP bug #78265 documents the problem; hopefully, we will have this feature soon. In the meantime, we will do the best we can by carefully adjusting the monitor manually, using the controls built in to most quality monitors:

1. Set the monitor white point to 6,500K. This provides more yellow than the 9,300K that most monitors ship with, but it is much easier for your eyes to see correct color, which is why it's the industry standard recommendation.
2. Increase the monitor contrast to 100% and adjust the brightness until you can see most of the B Black Level strip in Norman Koren's excellent calibration chart, shown in Figure 3.

3. Adjust the display gamma using the gamma strip of Figure 3. To read the chart, squint your eyes until you can't discern the horizontal lines and find the location where the gray tone is the same across the strip; this is your current gamma. Adjust the display gamma using xgamma until the gamma is 2.2, the industry default.
4. Now adjust the separate R, G and B controls to make the gray of the target as neutral as possible. Use an image such as the color calibration chart, shown in Figure 3, to help adjust the monitor color balance. By focusing on the skin tones as well as neutral gray tones, you should be able to eliminate any gross color imbalances.

Figure 3. Monitor calibration images: on the left, Norman Koren's monitor gamma chart, and on the right, the Photodisc color calibration image. These printed examples should not be used for reference as their colors may have changed due to printing press inconsistencies.

In addition to a well-adjusted monitor, a consistent lighting environment is critical for accurate color viewing. Dim light is best, and by lighting your room with 5,000K daylight-balanced compact fluorescents, the white of a piece of paper should look very close to the white on your display. ISO 12646 goes into great depth on providing further guidelines on ambient viewing conditions. Bright colors in your environment alter your color perception, so watch out for brightly colored nearby walls, as well as bright backgrounds and colored windows in your Linux environment. My Linux desktop uses a plain gray background and black and white window decorations.

Scanner Calibration

Now that we have a reasonably correct monitor and viewing environment, we can turn our focus to scanning. Both SANE and VueScan are excellent Linux solutions for scanning, I have chosen to use VueScan primarily because it's the same software I use on the Mac. Whichever scanning software you choose, it's important always to use the same software settings. This is vital for correct color, as the ICC profile we are about to create is valid only for the color settings used when making it. In order to make a scanner profile, it's necessary to have a known quantity to scan, called a color calibration target. The target I use is the Q-60 Color Input Target made by Kodak, and it is available from many high-end photo stores. Scan the target using your preferred scanning software; remember to save your settings, and always use the same settings moving forward.

Creating an ICC from this scan is made possible by the LProf profiling tools, available from the Little CMS Web site (see on-line Resources). Run kmeasurementool and open the scan. Go to the Options tab and select the correct pick template for the target used, in this case the Kodak column picker. Go back to the image tab and resize the green target outline. Once the little squares are lined up with the color patches in the scan, select Pick and then Save IT8 sheet. Now use kscannerprofiler to create a custom ICC profile for your scanner by first selecting the correct target vendor and batch ID. Each target comes with a date identifying the batch. Open the IT8 sheet file created above and enter an output profile filename for your ICC profile, then press GO. This ICC profile now can be used to interpret the color space of the scanner.

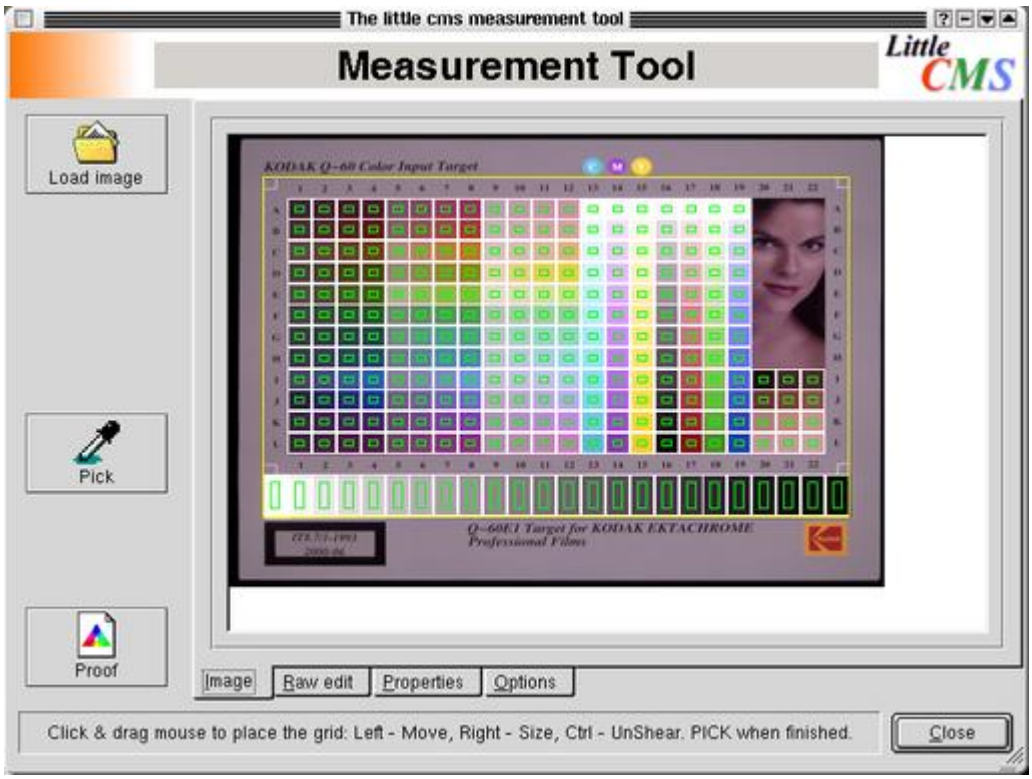


Figure 4. Creating the Scanner Profile Using Scanned Kodak Target

Now that we have profiled our scanner, we can scan an actual transparency. When scanning, it's helpful to know the true optical resolution of the scanner. Scanning at the maximum optical resolution captures as much detail as possible without making your file too large. When saving the file, use the TIFF format for best results as it is a lossless format.

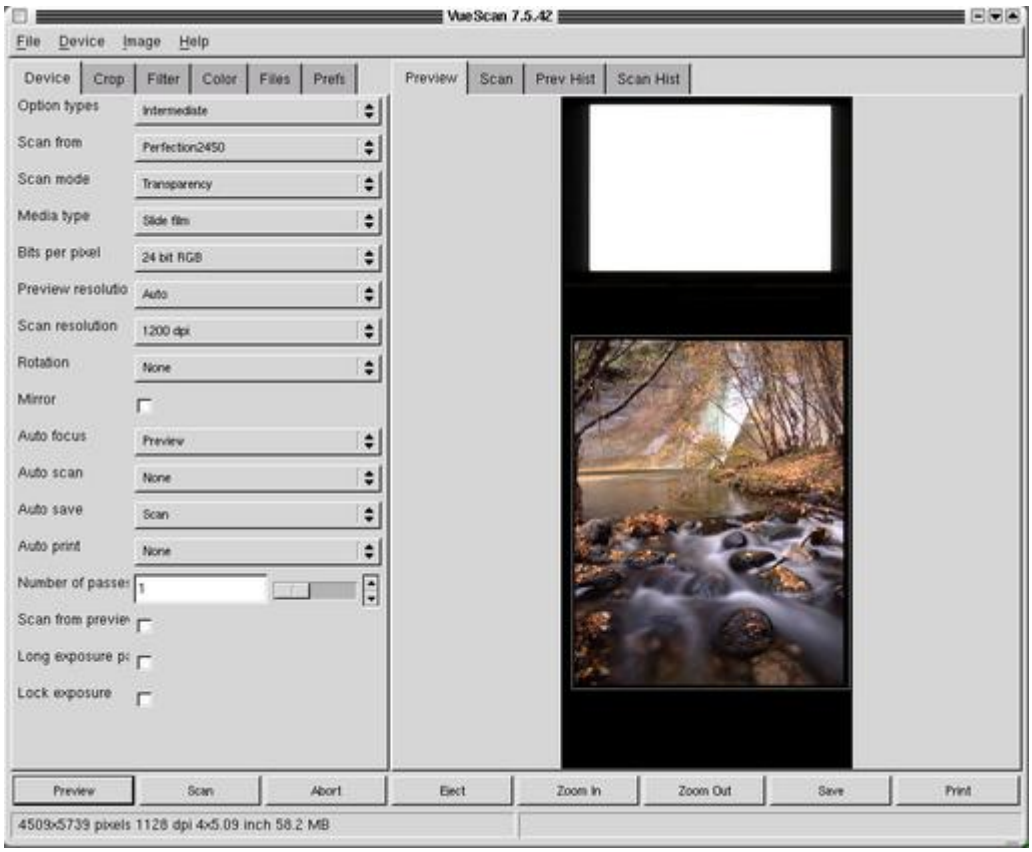


Figure 5. Making the Initial Scan Using VueScan

After scanning the transparency, the resulting TIFF file's color space is that of the scanner. I want to convert this color space to a working color space to do my image editing. A popular working color space is the Adobe 1998, a profile of which can be obtained from the Adobe Web site. Apply the scanner profile using tifficc, which is part of the Little CMS package, by running `tifficc -i scanner.icc -o Adobe1998.icc scan.tiff scan_working.tiff`, which converts the scanned TIFF image to our working color space.

Photographic Tools in The GIMP

A lot of the tools in The GIMP, as well as other image editing programs, aren't needed for photographic work. Most of my time is spent using the Levels or Curves tools. The first adjustment I make is to the overall exposure, and the best tool for this is Levels. Never use the Brightness/Contrast tool, as it tends to

clip values. Open the Levels tool and slide the black and white points right up to the point where data (height) is apparent. Adjust the middle slider to lighten or darken the overall image.

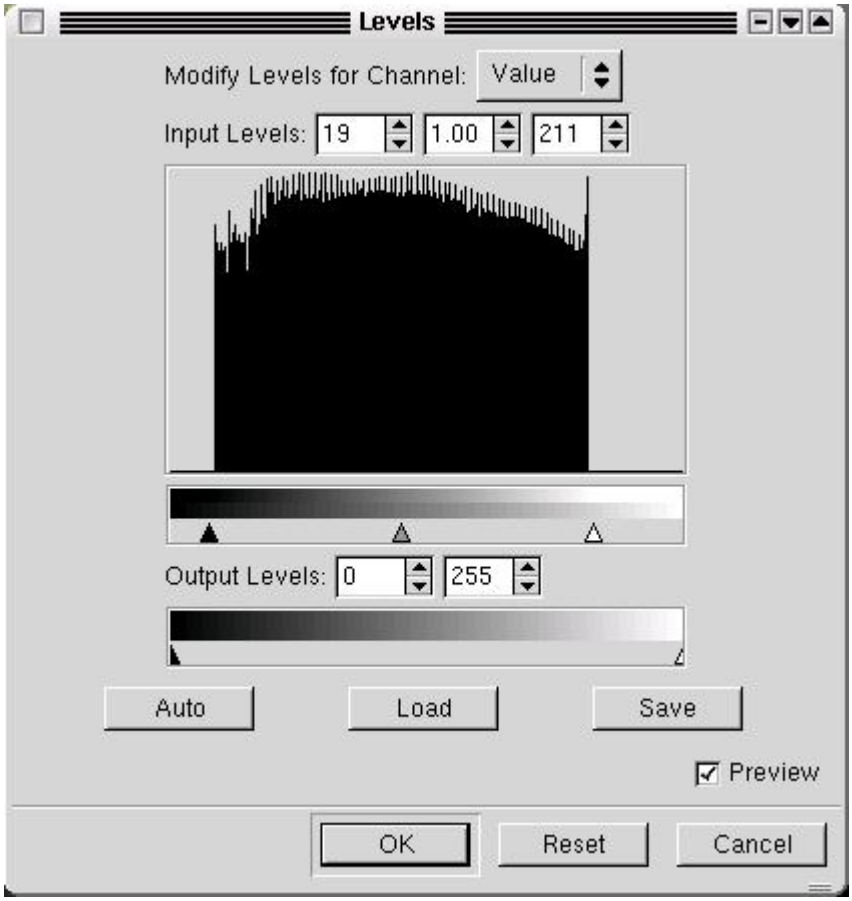


Figure 6. Using Levels in The GIMP for Exposure Adjustment

Cleaning up dust and scratches is the vital next step, and the Clone tool is perfect for this. Zoom in until the size of a dust spot is easily seen on the screen and then carefully scan the entire image. When you find a spot, select the Clone tool using a soft-edged brush that is slightly larger than the dust spot. Set the Clone Tool Options to 100% opacity and select aligned. Ctrl-click on an adjacent area to pick up an area to copy, release the Ctrl key and click directly on top of the spot. By using an opaque brush, the spot should be removed completely. The technique for removing scratches is similar, but you have to watch out for repeating patterns. With a little practice, you should be able to clone large sections without any telltale artifacts.

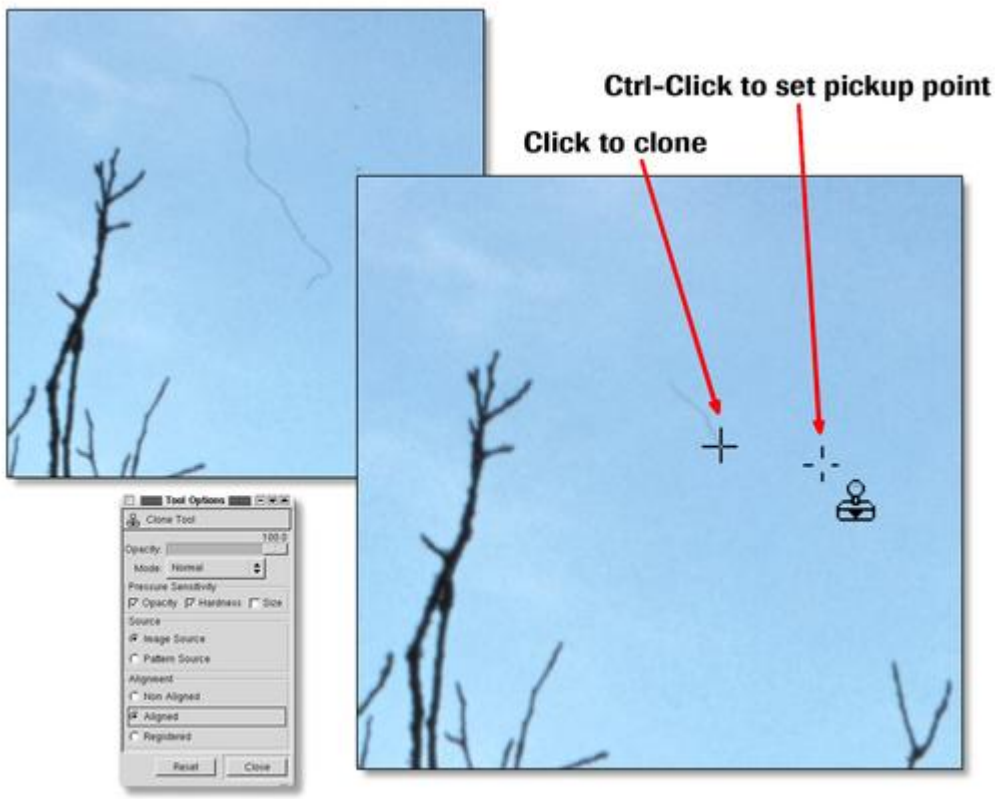


Figure 7. Using the Clone Tool to Remove Dust and Scratches

The Artist's Proof

We now are ready to make a print. For me, the print is the final piece of art, and I spend a lot of time trying to create a print that really speaks to the viewer. I often will print out a picture 10–20 times, subtly making adjustments until I am happy with the image. I call this inkjet print my artist proof.

To prepare the image for printing, a few adjustments must be made that are specific to the print size. These adjustments are destructive, so save a copy of your image to a master file before getting started. First, resize the image by opening up the Scale Image dialog under The GIMP's Image menu. For inkjets I recommend 200PPI at your desired print size. Next, sharpen the image using the Unsharp Mask tool. Think of this as enhancing the edges in a photo. In case you're wondering why it's called unsharp, the name comes from the traditional photographic method of using a blurred (or unsharp) copy negative to increase the edge definition, and thus the perceived sharpness, of an image. Select Filter→Enhance→Unsharp Mask, and a dialog box appears with three sliders. Radius is how far away from the edge pixels are affected, Amount controls how much the pixels are changed and Threshold determines which edges are affected. To decide how much sharpening to apply, view the image at 100% and try a few settings. Unfortunately, Unsharp Mask does not yet have a preview, so you have to apply a setting, undo and repeat until you get the look you want. I am looking for the setting where the sharpening is barely obvious on my

screen, generally starting with a radius of 1.0, a threshold of 3 and an amount of .75.

I use CUPS with Gimp-Print when printing to the Epson inkjets. Check the Gimp-Print Web page for instructions on getting this set up and running. By using Gimp-Print, all of the Epson-specific options are available in the Print menu. It also includes the Epson printer utility escputil, which allows for head cleaning and print alignment. Ideally, at this point we would be able to create an ICC profile for our printer; however, no software for Linux exists yet that does this. Instead, I recommend creating a standard set of adjustments using Gimp-Print's color adjust feature. To arrive at these adjustments, use a color test image such as the one we used for monitor calibration. Print the image, and look carefully at the results. You should be able to see details in the shadows and separation in the grayscale step tablet; the neutral and skin tones should not have a color bias. If this isn't the case, open the Print dialog box, select Image/Output Settings→Adjust Output and make the necessary changes. Repeat until your printout looks correct. Once you find a standard set of adjustments, write them down so you can use them as a starting point for future prints.

Figure 8. Using Gimp-Print's Color Adjust Dialog

The moment of truth has finally arrived—will the final print match what we have been viewing on the screen? Print out the image and compare it to the image onscreen. They should be very similar, but something is fundamentally different about looking at a print vs. the screen. Study the image and see where

improvements can be made, and continue making refinements in The GIMP until you are satisfied.

Working with a Service Bureau

For salable, fine-art prints, I prefer Lightjet prints made by a service bureau on a Cymbolic Sciences Lightjet printer. This printer uses lasers to expose the image onto conventional photo paper, most often Fuji Crystal Archive paper. Wilhelm Imaging Research tests (the authority on photo print life) show these prints have a display life of 60 years with little color shift or fading, significantly longer than most inkjet prints. These prints are indistinguishable from conventional photos and can be made as large as 4 × 5 feet.

Preparing the file for output on the Lightjet is similar to preparing it for inkjet printing. Start by going back to the original file that was saved, not the file for inkjet output. Again, resize the file to the correct physical dimensions, this time using 300PPI rather than 200PPI, sharpen the file using the technique described above and save it to a separate TIFF file. The service bureau I use, Calypso Imaging, offers Lightjet prints for photographers across the US. They also offer a discount if the file to be output has been resized and the correct ICC profile, available from their Web site, applied. Download the latest ICC profile and apply it to your image using tifficc. The image then is ready to be burned to CD-ROM or uploaded to Calypso's FTP site for final output.

A few days later I receive the final print, look back at the process used to create it and contemplate how, although my photo tools have changed, the goal remains the same—to create a print that conveys my feelings about my chosen subject. Finally, I am able to go through this entire process using Linux in combination with affordable desktop products, all in my own digital studio. Although a few areas still need improvements, Linux is very much up to the task. So get out there, take some photos and try out these Linux tools to make great photographic prints.

Resources for this article: /article/7704.

When RW Hawkins is not helping companies with their Linux servers, he is camping and backpacking with his 4 × 5 camera in the canyons of the Southwest. He lives in the Silicon Valley with his wife, who prefers pizza to camp food. His Web site, rwhawkins.com, offers a gallery of his fine-art prints as well as technical advice on digital imaging.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Porting RTOS Device Drivers to Embedded Linux

Bill Weinberg

Issue #126, October 2004

Transform your wild-and-woolly legacy RTOS code into well-formed Linux device drivers.

Linux has taken the embedded marketplace by storm. According to industry analysts, one-third to one-half of new embedded 32- and 64-bit designs employ Linux. Embedded Linux already dominates multiple application spaces, including SOHO networking and imaging/multifunction peripherals, and it now is making vast strides in storage (NAS/SAN), digital home entertainment (HDTV/PVR/DVR/STB) and handheld/wireless, especially in digital mobile phones.

New embedded Linux applications do not spring, Minerva-like, from the heads of developers; a majority of projects must accommodate thousands, even millions of lines of legacy source code. Although hundreds of embedded projects have successfully ported existing code from such platforms as Wind River's VxWorks and pSOS, VRTX, Nucleus and other RTOSes to Linux, the exercise is still nontrivial.

To date, the majority of literature on migration from legacy RTOS applications to embedded Linux has focused on RTOS APIs, tasking and scheduling models and how they map to Linux user-space equivalents. Equally important in the I/O-intensive sphere of embedded programming is porting RTOS application hardware interface code to the more formal Linux device driver model.

This article surveys several common approaches to memory-mapped I/O frequently found in legacy embedded applications. These range from ad hoc use of interrupt service routines (ISRs) and user-thread hardware access to the semi-formal driver models found in some RTOS repertoires. It also presents heuristics and methodologies for transforming RTOS code into well-formed Linux device drivers. In particular, the article focuses on memory mapping in RTOS code vs. Linux, porting queue-based I/O schemes and redefining RTOS I/O for native Linux drivers and dæmons.

RTOS I/O Concepts

The word that best describes most I/O in RTOS-based systems is informal. Most RTOSes were designed for older MMU-less CPUs, so they ignore memory management even when an MMU is present and make no distinction between logical and physical addressing. Most RTOSes also execute entirely in privileged state (system mode), ostensibly to enhance performance. As such, all RTOS application and system code has access to the entire machine address space, memory-mapped devices and I/O instructions. Indeed, it is very difficult to distinguish RTOS application code from driver code even when such distinctions exist.

This informal architecture leads to ad hoc implementations of I/O and, in many cases, the complete absence of a recognizable device driver model. In light of this egalitarian non-partitioning of work, it is instructive to review a few key concepts and practices as they apply to RTOS-based software.

In-Line Memory-Mapped Access

When commercial RTOS products became available in the mid-1980s, most embedded software consisted of big mainline loops with polled I/O and ISRs for time-critical operations. Developers designed RTOSes and executives into their projects mostly to enhance concurrency and aid in synchronization of multitasking, but they eschewed any other constructs that got in the way. As such, even when an RTOS offered I/O formalisms, embedded programmers continued to perform I/O in-line:

```
#define DATA_REGISTER  0xF00000F5

char getchar(void) {
  return (*((char *) DATA_REGISTER));
}


void putchar(char c) {
  *((char *) DATA_REGISTER) = c;
}
```

More disciplined developers usually segregate all such in-line I/O code from hardware-independent code, but I have encountered plenty of I/O spaghetti as well. When faced with pervasive in-line memory-mapped I/O usage, embedded developers who are new to Linux always face the temptation to port all such code as-is to user space, converting the `#define` of register addresses to calls to `mmap()`. This approach works fine for some types of prototyping, but it cannot support interrupt processing, has limited real-time responsiveness, is not particularly secure and is not suitable for commercial deployment.

RTOS ISRs

In Linux, interrupt service is exclusively the domain of the kernel. With an RTOS, ISR code is free-form and often indistinguishable from application code, other than in the return sequence. Many RTOSes offer a system call or macro that lets code detect its own context, such as the Wind River VxWorks intContext(). Common also is the use of standard libraries by ISRs, with accompanying reentrancy and portability challenges.

Most RTOSes support the registration of ISR code and handle interrupt arbitration and ISR dispatch. Some primitive embedded executives, however, support only direct insertion of ISR start addresses into hardware vector tables. Even if you attempt to perform read and write operations in-line in user space, you have to put your Linux ISR into kernel space.

RTOS I/O Subsystems

Most RTOSes ship with a customized standard C run-time library, such as pREPC for pSOS, and selectively patched C libraries (libc) from compiler ISVs. They do the same for glibc. Thus, at a minimum, most RTOSes support a subset of standard C-style I/O, including the system calls open, close, read, write and ioctl. In most cases, these calls and their derivatives resolve to a thin wrapper around I/O primitives. Interestingly, because most RTOSes did not support filesystems, those platforms that do offer file abstractions for Flash or rotating media often use completely different code and/or different APIs, such as pHILE for pSOS. Wind River VxWorks goes further than most RTOS platforms in offering a feature-rich I/O subsystem, principally to overcome hurdles in integration and generalization of networking interfaces/media.

Many RTOSes also support a bottom-half mechanism, that is, some means of deferring I/O processing to an interruptible and/or preemptible context. Others do not but may instead support mechanisms such as interrupt nesting to achieve comparable ends.

Typical RTOS Application I/O Architecture

A typical I/O scheme (input only) and the data delivery path to the main application is diagramed in Figure 1. Processing proceeds as follows:

- A hardware interrupt triggers execution of an ISR.
- The ISR does basic processing and either completes the input operation locally or lets the RTOS schedule deferred handling. In some cases, deferred processing is handled by what Linux would call a user thread, herein an ordinary RTOS task.

- Whenever and wherever the data ultimately is acquired (ISR or deferred context), ready data is put into a queue. Yes, RTOS ISRs can access application queue APIs and other IPCs—see the API table.
- One or more application tasks then read messages from the queue to consume the delivered data.

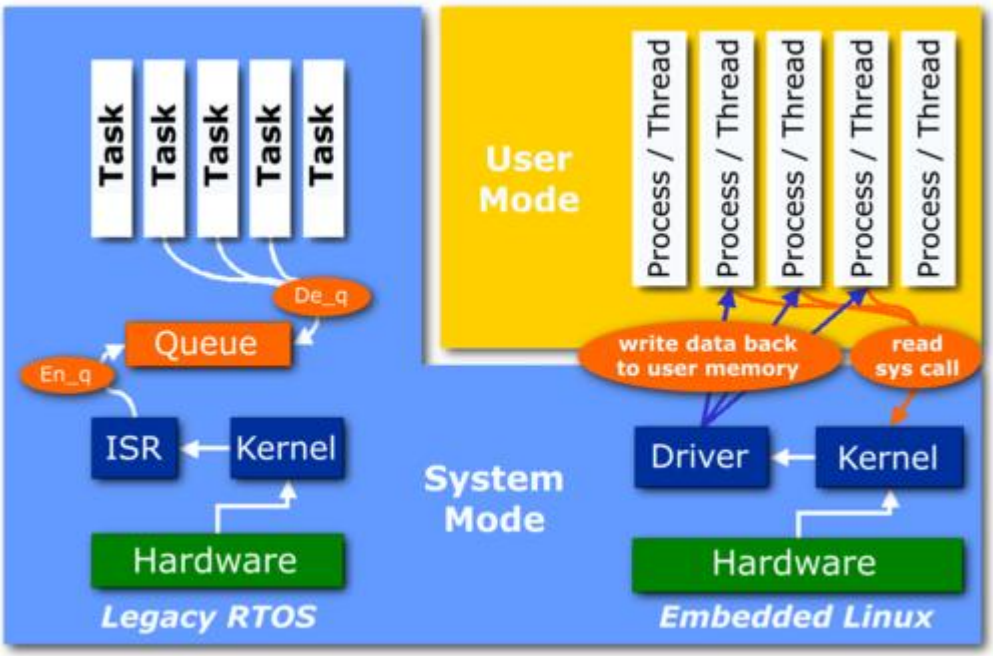


Figure 1. Comparison between Typical I/O and Data Delivery in a Legacy RTOS and Linux

Output often is accomplished with comparable mechanisms—instead of using write() or comparable system calls, one or more RTOS application tasks put ready data into a queue. The queue then is drained by an I/O routine or ISR that responds to a ready-to-send interrupt, a system timer or another application task that waits pending on queue contents. It then performs I/O directly, either polled or by DMA.

Mapping RTOS I/O to Linux

The queue-based producer/consumer I/O model described above is one of many ad hoc approaches employed in legacy designs. Let us continue to use this straightforward example to discuss several possible (re)implementations under embedded Linux.

Developers who are reticent to learn the particulars of Linux driver design, or who are in a great hurry, likely try to port most of a queue-based design intact to a user-space paradigm. In this driver-mapping scheme, memory-mapped physical I/O occurs in user context by way of a pointer supplied by mmap():

```
#include <sys/mman.h>
```

```
#define REG_SIZE   0x4   /* device register size */
#define REG_OFFSET 0xFA400000
                   /* physical address of device */

void *mem_ptr;
         /* de-reference for memory-mapped access */
int fd;

fd=open("/dev/mem",O_RDWR);
           /* open physical memory (must be root) */

mem_ptr = mmap((void *)0x0,
               REG_AREA_SIZE, PROT_READ+PROT_WRITE,
               MAP_SHARED, fd, REG_OFFSET);
                         /* actual call to mmap() */
```

A process-based user thread performs the same processing as the RTOS-based ISR or deferred task would. It then uses the SVR4 IPC msgsnd() call to queue a message for receipt by another local thread or by another process by invoking msgrcv().

Although this quick-and-dirty approach is good for prototyping, it presents significant challenges for building deployable code. Foremost is the need to field interrupts in user space. Projects such as DOSEMU offer signal-based interrupt I/O with SIG (the silly interrupt generator), but user-space interrupt processing is quite slow—millisecond latencies instead of tens of microseconds for a kernel-based ISR. Furthermore, user-context scheduling, even with the preemptible Linux kernel and real-time policies in place, cannot guarantee 100% timely execution of user-space I/O threads.

It is highly preferable to bite the bullet and write at least a simple Linux driver to handle interrupt processing at kernel level. A basic character or block driver can field application interrupt data directly in the top half or defer processing to a tasklet, a kernel thread or to the newer work-queue bottom-half mechanism available in the 2.6 kernel. One or more application threads/processes can open the device and then perform synchronous reads, just as the RTOS application made synchronous queue receive calls. This approach will require at least recoding consumer thread I/O to use device reads instead of queue receive operations.

To reduce the impact of porting to embedded Linux, you also could leave a queue-based scheme in place and add an additional thread or dæmon process that waits for I/O on the newly minted device. When data is ready, that thread/dæmon wakes up and queues the received data for use by the consuming application threads or processes.

Porting Approaches

Porting RTOS code to embedded Linux does not differ conceptually from enterprise application migration. After the logistics of porting have been

addressed (make/build scripts and methods, compiler compatibility, location of include files and so on), code-level porting challenges turn on the issues of application architecture and API usage.

For the purposes of the discussion at hand, let us assume that the application part (everything except I/O-specific code) migrates from the RTOS-based system into a single Linux process. RTOS tasks map to Linux threads and intertask IPCs map to Linux inter-process and inter-thread equivalents.

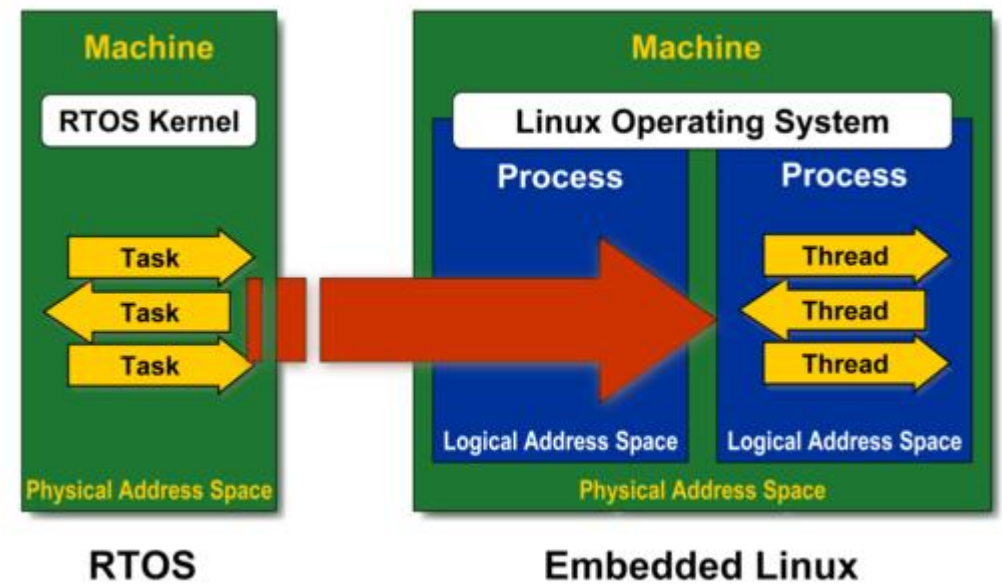


Figure 2. Mapping RTOS Tasks to Linux Process-Based Threads

Although the basic shape of the port is easy to understand, the devil is in the details. And the most salient details are the RTOS APIs in use and how to accommodate them with Linux constructs.

If your project is not time-constrained, and if your goal is to produce portable code for future project iterations, then you want to spend some time analyzing the current structure of your RTOS application and how/if it fits into the Linux paradigm. For RTOS application code, you want to consider the viability of one-to-one mapping of RTOS tasks onto Linux process-based threads and whether to repartition the RTOS application into multiple Linux processes. Depending on that decision, you should review the RTOS IPCs in use to determine proper intra-process vs. inter-process scope.

On the driver level, you definitely want to convert any informal in-line RTOS code to proper drivers. If your legacy application already is well partitioned, either using RTOS I/O APIs or at least segregated into a distinct layer, your task becomes much easier. If ad hoc I/O code is sprinkled liberally throughout your legacy code base, you've got your work cut out for you.

Developers in a hurry to move off a legacy RTOS or those trying to glue together a prototype are more likely to attempt to map or convert as many RTOS APIs to Linux equivalents in situ. Entities in common, such as comparable APIs, IPCs and system data types, port nearly transparently. Others can be addressed with #define redefinition and macros. Those remaining need to be recoded, ideally as part of an abstraction layer.

You can get a head start on API-based porting by using emulation libraries that accompany many embedded Linux distributions (including MontaVista's libraries for Wind River VxWorks and pSOS) or by using third-party API-mapping packages from companies such as MapuSoft.

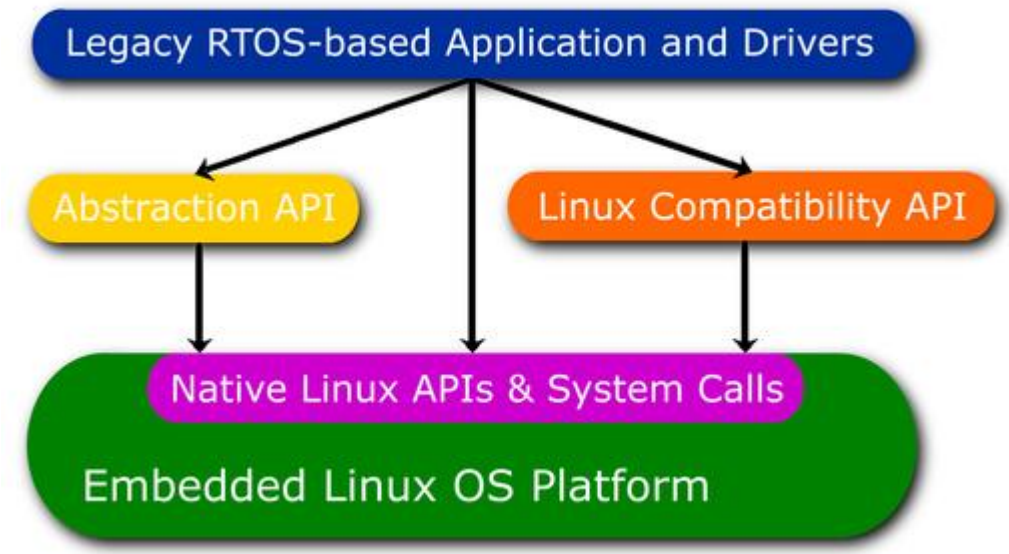


Figure 3. Multipronged Approach to Porting RTOS Code and APIs to Linux

Most projects take a hybrid approach, mapping all comparable or easily translatable APIs, re-architecting where it doesn't slow things down and playing Whack-a-Mole with the remaining code until it builds and runs.

Available APIs in Kernel and User Space

For both intensive re-architecting and for quicker-and-dirtier API approaches, you still have to (re)partition your RTOS application and I/O code to fit the Linux kernel and user-space paradigm. Table 1 illustrates how Linux is much stricter about privileged operations than a legacy RTOS and helps guide you in the (re)partitioning process.

Table 1. Privileged Operations in Linux and Legacy RTOSes

	IPCs	Synchronization	Tasking	Namespace
RTOS Application	Queues, Signals, Mailboxes Informal Shared Memory	Semaphores, Mutexes	Full RTOS Tasking Repertoire	Full Application, Libraries and System (Link-Time)
RTOS Driver	Queues, Signals, Mailboxes Informal Shared Memory	Semaphores, Mutexes	Full RTOS Tasking Repertoire	Full Application, Libraries and System (Link-Time)
Linux Application	Queues, Signals, Pipes Intra-Process Shared Memory Shared System Memory	Semaphores, Mutexes	Process and Threads APIs	Local Process, Static and Shared Libraries
Linux Driver (Static)	Shared System Memory Read/Write Process Memory	Kernel Semaphores Spinlocks	Kernel Threads, Tasklets	Full Kernel
Linux Module (Dynamic)	Shared System Memory Read/Write Process Memory	Kernel Semaphores Spinlocks	Kernel Threads, Tasklets	Module-Local and Exported Kernel Symbols

Two important distinctions are called out in Table 1:

- RTOSes are egalitarian, letting application and I/O code touch any address and perform almost any activity, whereas Linux is much more hierarchical and restrictive.

- Legacy RTOS code can see every symbol or entry point in the system, at least at link time, whereas Linux user code is isolated from and built separately from kernel code and its accompanying namespace.

The consequences of the Linux hierarchy of privileged access is normally only kernel code (drivers) actually accesses physical memory. User code that also does so must run as root.

In general, user-space code is isolated from the Linux kernel and can see only explicitly exported symbols as they appear in /proc/ksyms. Moreover, visible system calls to the kernel are not invoked directly but by calls to user library code. This segregation is intentional, enhancing stability and security in Linux.

When you write a driver, the opposite is true. Statically linked drivers are privy to the entire kernel namespace, not only exports, but have zero visibility into user-space process-based symbols and entry points. And, when you encapsulate driver code in run-time loadable modules, your program can leverage only those interfaces explicitly exported in the kernel by the *EXPORT_SYMBOL* macro.

Migrating Network Drivers

As indicated above, porting character and block device drivers to Linux is a straightforward if time-consuming activity. Porting network drivers, though, can seem much more daunting.

Remember that while Linux grew up with TCP/IP, most RTOSes had networking grafted onto them in the late 1990s. As such, legacy networking often only presents bare-bones capabilities, such as being able to handle only a single session or instance on a single port or to support only a physical interface on a single network medium. In some cases, networking architecture was generalized after the fact, as with Wind River VxWorks MUX code to allow for multiple interfaces and types of physical connection.

The bad news is that you likely have to rewrite most or all of your existing network interfaces. The good news is that re-partitioning for Linux is not hard and you have dozens of open-source network device driver examples to choose from.

Your porting task is to populate the areas at the bottom of Figure 4 with suitable packet formatting and interface code.

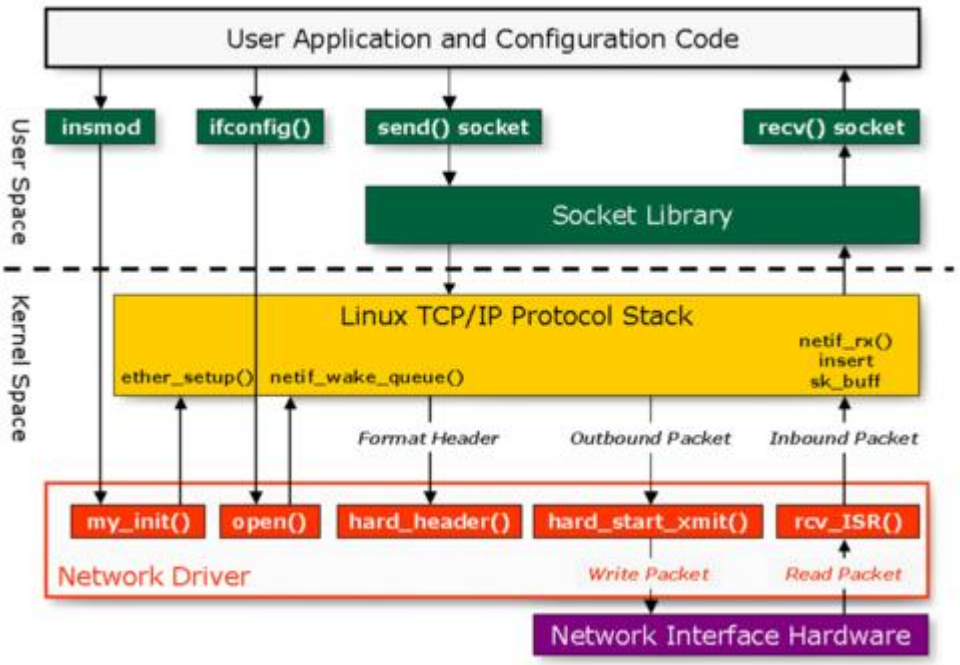


Figure 4. Block Diagram of Linux Network Drivers

Writing network drivers is not for beginners. Because, however, many RTOS network drivers actually were derived from existing GPL Linux interfaces, you might find the process facilitated by the code itself. Moreover, there is a large and still-growing community of integrators and consultants focused on making a business of helping embedded developers move their applications to Linux, for reasonable fees.

Conclusion

The goal of this article has been to give embedded developers some insight into both the challenges they will face and benefits they will realize from moving their entire software stack from a legacy RTOS to Linux. The span of 2,800 words or so is too brief to delve into many of the details of driver porting (driver APIs for bus interfaces, address translation and so on), but the wealth of existing open-source GPL driver code serves as both documentation and a template for your migration efforts. The guidelines presented here should help your team scope the effort involved in a port of RTOS to Linux and provide heuristics for re-partitioning code for the best native fit to embedded Linux.

As Director of Strategic Marketing and Technology Evangelist when he wrote this article, Bill focused his 17+ years of industry experience on advancing MontaVista and embedded Linux in today's dynamic pervasive computing marketplace. His background includes extensive embedded and real-time experience with expertise in OS, tools, software licensing and manufacturing.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

At the Forge

Syndication with RSS

Reuven M. Lerner

Issue #126, October 2004

Syndication isn't only for blogs. Use RSS to keep readers informed of changes to any Web site or application.

When I first started to use the Web, anyone who put up a site would send e-mail to Tim Berners-Lee, giving the URL and a brief description of what the site was about. Tim would respond with a brief personal note and would update his master list of Web sites, which anyone with a browser could retrieve. Active participants in the Web community would review that list regularly—and its successor, published by the same people who produced the Mosaic browser—for new and updated sites, so as not to miss a bit.

Fast-forward more than a decade, and the Web is obviously too large for anyone to maintain a list of new sites manually. And even if that were possible, no one can read more than a fraction of the new content that goes live each day. Add to this the fact that now there are hundreds of thousands of Weblogs, or blogs, many of which are frequently updated, and the task becomes even more difficult.

One solution is to use your browser's bookmarks. But after a while, it becomes a chore to check bookmarks each day, let alone several times a day. It would be nice if each site could indicate when its content has changed, so that you would visit only when necessary.

This insight is not new; the idea of announcing changes to Web content has existed for several years. But I must admit, it was only a few months ago when I began to realize how behind the times I was, when I would start each day by visiting a few of the sites in my bookmarks. By taking advantage of an RSS aggregator—namely, a program that looks at the RSS feeds from various sites

and alerts me when there has been an update—I am able to do more in less time.

This month, we discuss the popular RSS (really simple syndication or RDF site summary) family of formats, looking at ways in which it might be useful and how it is created.

Simple RSS

RSS began as the brainchild of Netscape, the Internet software company that has since been absorbed (and largely dismembered) by AOL. Netscape wanted to offer people news from multiple sources but on a single page. They accomplished this by publishing the specification for RSS 0.90. Anyone interested in publishing news through the Netscape portal needed to do so in RSS. Netscape's system would retrieve this RSS document from the Web site in question and publish the results.

Although RSS 0.90 sparked a revolution, it also was fairly complicated. Dave Winer, then the head of Userland Software, turned RSS into a simple specification, renamed it RSS 0.91 and began to talk about it on his Weblog, scripting.com. Suddenly, RSS 0.91 was everywhere; Dave's orange XML buttons, indicating that you could get an RSS feed from a site, became quite popular. Within a few years, RSS feeds sporting other versions were available as well. RSS 1.0 was developed by a group of developers on the Web, and RSS 2.0, coordinated by Dave, was seen as an upgrade to 0.9x.

If you have been following this history, you might have reached the conclusion that now there are three different syndication formats called RSS. Aside from the version numbers, and some obvious similarity between the different versions, these are three different formats.

In many ways, RSS resembles HTML and HTTP, which began as simple-to-understand, simple-to-implement standards written by a small group of people. All three of these standards have been forced to mature quite a bit in the past few years, losing some of their flexibility and simplicity in the process.

RSS 0.91 is the simplest of the bunch and is still rather popular. Everything sits within an <rss> element, which identifies its version and contains a single <channel> element. Several required tags (title, link, description, language and image) are followed by one or more <item> elements. Each item has its own title, link and description. For example, here is a simple RSS feed from my Weblog:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rss PUBLIC
```

```
"-//Netscape Communications//DTD RSS 0.91//EN"
"http://my.netscape.com/publish/formats/rss-0.91.dtd">

<rss version="0.91">

<channel>
  <title>Altneuland</title>
  <link>http://altneuland.lerner.co.il/</link>
  <description>Reuven's Weblog</description>
  <item>
    <title>Independence Day</title>
    <link>http://altneuland.lerner.co.il//40</link>
  </item>
  <item>
    <title>Linux desktops for the masses?  Ha!</title>
    <link>http://altneuland.lerner.co.il//39</link>
  </item>
</channel>
</rss>
```

If you examine the above RSS feed, you can see it does not conform to the RSS 0.91 specification I described previously. Specifically, it lacks the required language and image elements within channel, and it lacks a description element within each item. Unfortunately, this comes as no surprise; as was the case with HTML in its earliest years, software authors often cut corners, producing output that was good enough for most purposes. And indeed, COREBlog (which, as of this writing, I am using to produce my Weblog) seems to have cut such a corner, producing a usable, but substandard, RSS 0.91 feed.

Producing a Feed

If you want to produce a legitimate RSS feed, you probably should use one of the many open-source modules available for most popular languages. For example, Perl developers can use the XML::RSS module, available from any CPAN mirror (see the on-line Resources section).

To create an RSS feed with this module, we can write a simple program that looks like this:

```
#!/usr/bin/perl

use strict;
use diagnostics;
use warnings;

use XML::RSS;

my $url = "http://altneuland.lerner.co.il/";

my $rss = new XML::RSS (version => '0.91');
$rss->channel(title         => 'Altneuland',
      link          => $url,
      language      => 'en',
      description   => "Reuven Lerner's Weblog");

$rss->add_item(title => 'Being scared',
       link => "$url/43/index_html",
       description => 'Blog entry'
      );

print $rss->as_string;
```

We begin the program with the creation of a new XML::RSS object, specifying that we want to use version 0.91 of the RSS standard. We then specify the individual items we want to define, and we can omit the image tag. Although the XML::RSS module allows us to omit any or all of the tags from our channel descriptor, it would not make sense to leave out some of them, such as title and link.

We then add individual items to the channel, one by one, until we have completed all of them. At that point, we can produce the RSS output, which appears as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rss PUBLIC
"-//Netscape Communications//DTD RSS 0.91//EN"
"http://my.netscape.com/publish/formats/rss-0.91.dtd">

<rss version="0.91">

<channel>
<title>Altneuland</title>
<link>http://altneuland.lerner.co.il/</link>
<description>Reuven Lerner's Weblog </description>
<language>en</language>

<item>
<title>Being scared</title>
<link>http://altneuland.lerner.co.il/43/index_html</link>
<description>Blog entry</description>
</item>

</channel>
</rss>
```

Most programs that produce RSS feeds are not going to invoke $rss->add_item(), as I did above, on a case-by-case basis. If we are syndicating a Weblog, commercial news feed or other frequently updated site, we probably would create an RSS feed by looping over a set of files in a directory or (better yet) over rows in a relational database.

For example, the following code fragment would retrieve all of the Weblog entries posted within the last 24 hours to a hypothetical weblog_entries table in PostgreSQL:

```
# Get all entries from the latest 24 hours
my $sql = "SELECT entry_id, title, link, description
     FROM weblog_entries
    WHERE when_entered >= (NOW() - interval '1 day')";

# Prepare the SQL statement
my $sth = $dbh->prepare($sql);

# Execute the SQL statement
my $result = $sth->execute;

# Iterate through resulting rows
while (my $rowref = $sth->fetchrow_arrayref)
{
my ($id, $title, $link, $description) = @$rowref;
```

```
    $rss->add_item(title => $title,
                   link => $link,
                   description => $description
                  );
    }
```

This demonstrates one of the many advantages of storing a Weblog in a relational database. Once the entries are stored in a database, it is easy to add new functionality, such as syndication. Although XML::RSS provides functionality (and sample code, in its perldoc on-line documentation) for limiting the number of syndicated articles to a set number, this seems like a much more appropriate job for a database, where the LIMIT modifier can set a maximum number of returned rows.

Moving to RSS 1.0

RSS 1.0 was a reaction to RSS 0.91, tying it more closely to various World Wide Web Consortium (W3C) standards, including RDF. The version numbers might have you believe that 1.0 was an upgrade to 0.91; however, the two are (unfortunately) independent and uncoordinated. 0.91 (and its successor, RSS 2.0) have been authored by Dave Winer based on input from the developer community, and 1.0 was written by an open consortium of developers. RSS 0.91 and 2.0 have more in common than 1.0 does with either of them, which, not surprisingly, has led to some confusion.

RDF, the resource development framework defined by the W3C, is part of the semantic Web project, which wants to make the Web understandable to computers as well as by people. This requires standardizing the metadata, or invisible descriptions that accompany the output from a site. RDF is one attempt at such a standardization.

RSS 1.0 thus tied syndication to RDF, adding the use of XML namespaces along the way. XML namespaces allow us to combine different XML definitions into a single document.

To create a syndication feed that complies with RSS 1.0, we need to make only a simple change to our program from above, changing the version number in our invocation of new on XML::RSS:

```
    my $rss = new XML::RSS (version => '1.00');
```

And indeed, if we make this change, the resulting RSS feed looks slightly different:

```
    <?xml version="1.0" encoding="UTF-8"?>

    <rdf:RDF
```

```
 xmlns="http://purl.org/rss/1.0/"



>

<channel rdf:about="http://altneuland.lerner.co.il/">
<title>Altneuland</title>
<link>http://altneuland.lerner.co.il/</link>
<description>Reuven Lerner's Weblog </description>
<dc:language>en</dc:language>
<items>
 <rdf:Seq>
  <rdf:li rdf:resource=
   "http://altneuland.lerner.co.il/43/index_html" />
 </rdf:Seq>
</items>
</channel>

<item rdf:about=
 "http://altneuland.lerner.co.il/43/index_html">
<title>Being scared</title>
<link>http://altneuland.lerner.co.il/43/index_html</link>
<description>Blog entry</description>
</item>

</rdf:RDF>
```

There are several things to notice in this output, beginning with the definition and use of several namespaces, introduced with the xmlns attributes, and then with the use of additional RDF-specific attributes, such as rdf:about and rdf:resource.

But the above doesn't do justice to RSS 1.0, which allows us to specify a great number of other parameters. For example, we can set information about our site's frequency of syndication updates by adding a syn section to our invocation of $rss->channel(); RSS 1.0 also includes support for Dublin Core, an increasingly popular and standard method for tagging documents.

RSS 2.0

The good news, as we have seen, is that RSS 1.0 is not significantly more difficult to create or parse than RSS 0.91 was, assuming you are using decent tools. However, the complexity of RSS 1.0 was seen as unnecessary by some.

Indeed, after several attempts to reach a consensus on RSS 1.0, a number of developers banded together to work on something now known as Atom. Although discussion of Atom will have to wait until next time, this spurred the RSS camp (led by Winer) to produce RSS 2.0.

You can produce RSS 2.0-compatible feeds by changing the version number in the invocation of new:

```
my $rss = new XML::RSS (version => '2.0');
```

You must say 2.0; neither 2 nor 2.00 will work, because the version check uses a string comparison, rather than a numeric one.

What does RSS 2.0 look like? Well, you might be surprised:

```
<?xml version="1.0" encoding="UTF-8"?>

<rss version="2.0"
 xmlns:blogChannel=
 "http://backend.userland.com/blogChannelModule">

<channel>
<title>Altneuland</title>
<link>http://altneuland.lerner.co.il/</link>
<description>Reuven Lerner's Weblog </description>
<language>en</language>

<item>
<title>Being scared</title>
<link>http://altneuland.lerner.co.il/43/index_html</link>
<description>Blog entry</description>
</item>

</channel>
</rss>
```

This looks a lot like RSS 0.91 and, thus, seems like a stripped-down version of RSS 1.0. But when we remember that RSS 2.0 is a successor to 0.91 and is designed to fix some of its flaws while remaining small, simple to implement and flexible, it becomes more obvious.

RSS 2.0 includes a number of improvements over 0.91, most especially the idea of using namespaces as modules that add new functionality. RSS 2.0 doesn't define or use nearly as many namespaces as 1.0 does, but that's because it is not trying to implement RDF.

Partly because of criticism that he personally held the copyright to the RSS 2.0 specifications, Winer gave ownership to Harvard University. It is assumed that Winer will continue to play a major role in the development of RSS 2.0, but he will no longer be the final arbiter regarding usage or extensions.

However, the split seems to be final; there is now an Atom camp and an RSS camp, and I find it hard to believe they will meet. But given the conflicting goals they have set for themselves, this should not come as a surprise—after all, you cannot expect to have flexibility and ease of implementation in the same specification.

Conclusion

This month, we looked at the different flavors of RSS currently in use and compared their different styles and project goals. Luckily, someone who wants

to produce a bare-bones syndication feed does not need to work very hard. Although programmers can add some version-specific fields, the basics are the same for all versions of RSS, even those that are ostensibly incompatible. The resulting RSS feeds, of course, can look quite different, depending on which version is used.

Next column, we will look at the upstart Atom syndication format, which is growing rapidly as a competitor to RSS. Once we have done that, we will look at how to build our own news aggregator, allowing us to interpret and work with syndication feeds from a wide variety of sources. We also will consider different ways in which RSS can be used and how aggregators can provide more than the latest news and opinions.

Resources for this article: /article/7702.

Reuven M. Lerner, a longtime Web/database consultant and developer, now is 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

Filesystem Labeling in SELinux

James Morris

Issue #126, October 2004

SELinux needs to store extra security information about each file, and Linux makes it possible with extended attributes.

With NSA Security-Enhanced Linux now integrated into the 2.6 kernel and making its way into distributions, an increasing number of people likely will be installing SELinux and experimenting with it. Given this increasing user base, this article takes a closer look at filesystem labeling under SELinux. This is an intermediate-level article. Although a brief review of some SELinux concepts is provided in the next section, pointers to more detailed information can be found in the on-line Resources section. In particular, Faye Coker's introductory article [*LJ*, August 2003] is recommended as a starting point.

SELinux Overview: Labeling and Access Control

In SELinux, important objects, such as tasks, inodes and files are assigned a security context, a label that encapsulates the security attributes associated with an object. Under standard SELinux, this label is a colon-separated ASCII string composed of values for identity, role and type.

These labels are assigned by a kernel component called the security server, using rules loaded into the security policy database. The security context label on my workstation's /etc/shadow file is:

```
system_u:object_r:shadow_t
```

where `system_u` and `object_r` are generic identity and role values used for files. shadow_t is the type of the file, an attribute that determines how the file can be accessed. A process, for example, would be labeled like this:

```
root:staff_r:staff_t
```

The SELinux identity here is root, assigned by SELinux as a more permanent form of identity than the standard UNIX identity. The role is staff_r, indicating that the process has all of the permissions assigned to that role. The type assigned to the process is staff_t. For a process, the type attribute indicates how it is allowed to access objects and interact with other processes. The type attribute of a process often is referred to as a domain.

Access Control Decisions

SELinux uses these security context labels to make access control decisions between processes and objects, but how does this happen? SELinux has hooks located at strategic points within the core kernel code, such as the point where a file is about to be read by a user. These hooks allow SELinux to break out of the normal flow of the kernel to request extended access control decisions. Access control decisions usually are made between a process (for example, cat) and an object (for example, /etc/shadow) for a specific permission (read).

Decision requests are sent to the access vector cache (AVC), which passes requests through to the security server for interpretation. The security server consults the security policy database and determines a result, which is cached in the AVC and returned to the SELinux hook.

The SELinux hook then allows the flow to continue or return EACCES, depending on the decision result. Security context labels assigned to processes and objects are used to make these access control decisions.

Refer to Figure 1 for a simplified view of this process.

The following code demonstrates how security context labels are used on a real system, as seen by a user:

```
$ id -Z
root:staff_r:staff_t

$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

The audit log records the following:

```
avc:  denied  { read } for  pid=13653 exe=/bin/cat
name=shadow dev=hda6 ino=1361441 scontext=root:staff_r:staff_t
tcontext=system_u:object_r:shadow_t tclass=file
```

Translation: the cat program, labeled with the security context `root:staff_r:staff_t`, was denied permission to read a file labeled `system_u:object_r:shadow_t`.

SELinux knows nothing about the meaning of cat or /etc/shadow. It is concerned only with their respective security context labels, the class of the target object (in this case, it's a file) and the permission being requested.

An important aspect of SELinux design is that labels encapsulate all security attributes of an object, and they are interpreted only by the security server in the kernel and by libselinux in user space. The rest of the kernel code and user space merely pass labels around as opaque data. New security attributes can be added to labels without having to recompile applications or redesign core SELinux code.

Extended Attributes

On a typical Linux disk-based filesystem, each file is identified uniquely by an inode containing critical metadata for the file, including UNIX ownership and access control information. When the kernel references a file, its inode is read from disk into memory. A standard UNIX permission check simply uses the information present within the inode. SELinux extends standard UNIX security and uses security context labels to make extended access control decisions.

Linux implements Extended Attributes, also called EAs or xattrs. These are name/value pairs associated with files as an extension to normal inode-based attributes. EAs allow functionality to be added to filesystems in a standardized way so that interfaces to the attributes are filesystem-independent. Examples of EA functionality are access control lists (ACLs), storage of character-set metadata alongside file data and SELinux security context labeling.

EAs are stored within namespaces, allowing different classes of EAs to be managed separately. ACLs are stored in the system.posix_acl_access and system.posix_acl_default namespaces. SELinux security context labels are stored in the security.selinux namespace. See attr(5) for more information on EAs under Linux.

EA Security Labeling

EAs can be managed manually with the getfattr(1) and setfattr(1) utilities. For example, to view the SELinux security context label of a file:

```
$ getfattr -n security.selinux /tmp/foo
getfattr: Removing leading '/' from absolute
path names
# file: tmp/foo
security.selinux="root:object_r:sysadm_tmp_t\000"
```

Notice the specification of the EA security namespace. A wrapper utility called getfilecon(1) is provided for use with SELinux. It saves you from having to specify the EA namespace, and it has a cleaner output.

The use of text-based labels ensures that meaningful, human-readable security attributes are stored along with the file data. These labels can be preserved or translated if the filesystem is mounted on a different system, possibly with a different security policy. A counter example is the way the owner of a file is stored as a numeric UID in the file's inode. The UID typically is mapped to a meaningful value by way of /etc/passwd; it may not have the same meaning on a different system.

For a filesystem to support SELinux security context labels, it needs EA support and a handler for the EA security namespace. Such filesystems currently include ext3, ext2, XFS and ReiserFS; the last uses an external patch. In addition, the devpts filesystem has a dummy security handler that allows EA-based access to the in-kernel labels of ptys.

So, when are files labeled? During an SELinux system installation, the setfiles(8) utility typically is used to label all of the files in filesystems that support EA security labeling. Package management tools such as RPM also may label files during installation, while system administrators often need to set security contexts manually with chcon(1) or setfilecon(1).

Evolution of SELinux Filesystem Labeling

The first release of SELinux in 2000 used a different mechanism for labeling filesystems than is used by the extended attributes approach discussed in this article. Persistent security IDs (PSIDs), integer representations of security context labels, were stored in an unused field of the ext2 inode. Mapping files on each filesystem were used by SELinux to look up a file's PSID by inode and then to map the PSID to a security context label.

This approach had the disadvantage of needing to modify each filesystem separately to support PSIDs. Thus, it was not a good general solution for extended security in the upstream kernel.

With the LSM Project, a generalized access control framework was implemented for the Linux kernel. As no filesystem-specific hooks are used in LSM, SELinux moved away from the modified filesystem approach and stored PSIDs in a normal file next to the mapping files. This allowed SELinux to be used purely as an LSM application with no kernel patching. It also allowed labeling to work for more filesystems, but it was not optimal in terms of performance and

consistency. Accessing files from within the kernel remains generally problematic.

As part of the process of merging SELinux into the mainline kernel, with more feedback from the community, SELinux moved to the current filesystem labeling model based on extended attributes. Extended attributes provide applications with a standard API, eliminating the need for custom system calls to manipulate security labels, while filesystems also can be used similarly by other security modules, even on the same filesystem, using the separation provided by EA namespaces.

File Creation

When a file is created, a matching rule in the security policy typically describes how to assign a label based on the security contexts of the parent directory and the current task. Here's an example:

```
$ id -Z
root:staff_r:staff_t
$ ls -dZ /tmp
drwxrwxrwt+ root  root system_u:object_r:tmp_t /tmp
$ touch /tmp/hello
$ getfilecon /tmp/hello
/tmp/hello      root:object_r:staff_tmp_t
```

In this case, the security policy contains a rule that states files created by staff_t in a directory labeled tmp_t must be labeled with the type staff_tmp_t. If there is no explicit rule, files are labeled with the context of the parent directory.

Privileged applications can override the above-stated rule by writing a security context to /proc/self/attr/fscreate. This security context then is used to label any newly created files. The setfscreatecon(3) library function encapsulates this operation.

Unlabeled files may exist if a filesystem has not been labeled properly before use or if files are created on a filesystem when SELinux is not enabled. In case of the latter, the SELinux kernel internally assigns a default context to unlabeled files for AVC calls, but it does not attempt to relabel them on disk. To restore a security context label manually, use restorecon(8).

An fsck-like utility is being developed for managing the scenario where unlabeled files have been created. To be run on boot, this utility will ensure that all files are labeled correctly before the system enters multiuser mode.

Labeling Behaviors

In the preceding section, we discussed file labeling for filesystems that both support EAs on disk and have handlers for the EA security namespace. When such a filesystem is mounted normally, it is said to use xattr labeling behavior.

When a filesystem is initialized by SELinux, such as when it is being mounted, a log message is generated that reads:

```
SELinux: initialized (dev hda6, type ext3), uses xattr
```

The `uses xattr` clause means the filesystem uses the xattr labeling behavior described above.

Many filesystems do not support EAs, and of those that do, not all have security namespace handlers. For on-disk filesystems, it may be that nobody has done the coding work yet or that EAs simply don't make sense for legacy filesystems such as vfat.

A proliferation of pseudo-filesystems have developed under Linux. Filesystems are becoming an increasingly favored user-kernel API mechanism. The most obvious of these is procfs, which is an interface between user space and various kernel components. Due to the long history of procfs, it has accumulated a lot of cruft, and new user-kernel filesystem APIs are encouraged to be implemented by way of separate filesystems. These filesystems are kernel resident and have no intrinsic EA support. Examples include usbfs, sysfs and selinuxfs.

Such non-EA cases are managed with a variety of labeling behaviors, according to rules in the security policy for each filesystem type.

The transition SIDs labeling behavior is used for devpts, tmpfs and shmfs filesystems. Files in these filesystems are labeled on demand in the kernel, based on the security contexts of the current task and a security context specified for the filesystem in policy.

devpts is a special-case transition SIDs filesystem. It provides EA API access to ptys by way of a dummy EA security handler. Privileged applications, such as sshd, use this feature to relabel ptys, overriding the transition SID labels.

The task SIDs labeling behavior simply labels the file with the same security context as the current task. It is used for pipes and sockets created in the pipefs and sockfs filesystems, respectively.

The genfs_contexts labeling behavior is used for filesystems unsuited to xattr, transition SIDs and task SIDs labeling. In the security policy, security context labels are assigned to filesystem/pathname pairs. The purpose of the pathname component is to allow finer-grained labeling of the filesystem. This feature is important for procfs in particular, which is a jumble of readable and writable kernel data, including the sysctl interface.

Most non-EA filesystems use genfs_contexts labeling, usually with the entire filesystem set to a single security context. Common examples include sysfs, vfat, nfs and usbdevfs.

Mountpoint Labeling

A new feature included with the 2.6.3 kernel is mountpoint labeling, also referred to as context mounts. The main purpose of this is to allow the security context of an entire filesystem to be specified by using a mount option. Mountpoint labeling can be applied to any type of filesystem and overrides its normal labeling behavior.

A specific use of mountpoint labeling is to allow different NFS mounts to be labeled separately at mount time. It also is useful for general ad hoc mounting of filesystems that do not support EA security labeling and for mounting EA-labeled filesystems labeled elsewhere. The latter may be important in forensic work, for example.

Legacy filesystems with no labels also may need to be mounted under an SELinux-enabled OS. Even though the filesystem type supports EA security labeling, we may not want to add persistent security context labels to these filesystems. Mountpoint labeling allows us to assign kernel-resident labels that are not written to disk.

As mountpoint labeling is a new feature and is not widely documented, let's discuss it in a little more detail.

When SELinux is enabled in the kernel, three new mount options are provided for mountpoint labeling:

- context: causes every file on the filesystem, and the filesystem itself, to be labeled with the specified security context. The /proc/self/attr/fscreate API discussed above is ignored for the filesystem. This overrides existing labeling behavior, changing it to mountpoint labeling. Filesystem labels are read-only to the user with this option, although policy-specified labeling transitions still operate on filesystems with EA security labeling support.

- fscontext: sets the label of the aggregate filesystem (that is, the filesystem itself) to the specified security context. This allows finer-grained control of filesystems by allowing their labels to be set on a per-mount basis rather than on a per-fs type basis specified in a policy. As the context option also implements this functionality, the two options cannot be used together. This option works only for filesystems with EA security labeling support. Aggregate filesystem security contexts are used for access control decisions made during file creation within a specific filesystem, mounting and unmounting of filesystems, accessing filesystem attributes and relabeling the filesystem itself.
- defcontext: sets the default security context for unlabeled files, instead of the value specified in the policy. As with the fscontext option, it works only for filesystems with EA labeling support and is not valid if context has been specified, as it too implements this functionality.

In the kernel, SELinux parses and strips out the security mount options during mount(2), passing normal options through to filesystem-specific code. Normal filesystems do not need to be aware of the security options, thus, they do not need to be modified. This is possible because most filesystems use text name/value pairs for mount options, which SELinux is able to manipulate easily.

Filesystems with binary mount option data, including NFS, SMBFS, AFS and Coda, need to be handled as special cases. Of these, only NFSv3 is supported at this stage of SELinux development.

Here's an example of how the context option operates, as it is likely to be the most widely used of the three mount options. A floppy disk with log files has arrived on our desk, and we'd like to mount it on our SELinux box and run some log analysis software on it. Due to the way policy is configured, these files need to be labeled system_u:object_r:var_log_t for the log analysis software to work properly. Mounting in this fashion also can help provide a sandbox for the data on the floppy, allowing SELinux to protect the OS and the contents of the floppy from each other.

Let's mount the disk:

```
$ mount -v -t vfat \
-o context=system_u:object_r:var_log_t \
/dev/fd0 /mnt/floppy
/dev/fd0 on /mnt/floppy type vfat
(context=system_u:object_r:var_log_t)
```

What does the audit log say?

```
SELinux: initialized (dev fd0, type vfat), uses mountpoint labeling
```

This message looks promising. Next, we verify that the files on the disk are labeled as expected. Normally, you would use getfilecon(1), but getfattr(1) has more explicit error messages:

```
$ getfattr -n security.selinux /mnt/floppy/access_log
/mnt/floppy/access_log: security.selinux: Operation not supported
```

What is going on here? An `ls -Z` also shows that the file has a null security context:

```
$ ls -Z /mnt/floppy/access_log
-rwxr-xr-x+  root  root  (null)   /mnt/floppy/access_log
```

The vfat filesystem on the floppy does not have EA support, and its security context labeling occurs purely within the kernel. It turns out that this in-kernel labeling is working correctly, but the user-space tools are not able to view the labels in the EA API. This is a limitation of the current EA implementation that has yet to be resolved elegantly.

However, there's a sneaky way to see what the labels on the files are by using the audit log, which always records the security context of a target object when logging an access message.

The use of getfattr(1) caused the following audit record to be generated:

```
avc:  denied  { getattr } for  pid=12354 exe=/usr/bin/getfattr
path=/mnt/floppy/access_log dev=fd0 ino=132 scontext=root:staff_r:staff_t
tcontext=system_u:object_r:var_log_t tclass=file
```

So, the file is labeled correctly (system_u:object_r:var_log_t), per the context mount option passed to the mount command.

Future Work

Although it is possible to assign security context labels to NFS mounted filesystems, they operate only locally for access control decisions within the kernel. No labels are transmitted across the network with files. Work has been advancing in this area, with SELinux-specific modifications being made to the NFSv2/v3 protocols and code. Further down the track, NFSv4 integration is expected to involve labeling over the wire by way of named attributes, which are part of the more extensible NFSv4 specification. This would allow both the NFS client and server to implement SELinux security for networked files. Support for other networked filesystems also would be useful, as would interoperability with Trusted BSD's SELinux port.

Backup and Restoration

One of the many tasks that change for system administrators using SELinux is backup and restoration. When creating an archive, how will the security context labels be preserved within the archive? The answer is to use the highly flexible star(1) utility, which has extended attribute support.

To manipulate archives with security context labels, use the xattr option. When creating archives, you also need to specify the exustar format. For example:

```
$ star -xattr -H=exustar -c -f cups-log.star /var/log/cups
```

creates an archive of the /var/log/cups directory, retaining security context labels on the files.

To extract, simply use the xattr option:

```
$ star -xattr -x -f cups-log.star
$ ls -Z var/log/cups/
-rw-r--r--+ root     sys      system_u:object_r:cupsd_log_t
error_log
-rw-r--r--+ root     sys      system_u:object_r:cupsd_log_t
error_log.1
```

As you can see, the security context labels have been preserved.

Resources for this article: /article/7689.

James Morris (jmorris@redhat.com) is a kernel hacker from Sydney, Australia, currently working for Red Hat in Boston. He is a kernel maintainer of SELinux, Networking and the Crypto API; an LSM developer and an Emeritus Netfilter Core Team member.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Cooking with Linux

The Game of Security

Marcel Gagné

Issue #126, October 2004

After dealing with the worm and virus traffic that a legacy OS dumps onto your network, you might feel like playing a game to relax.

Excellent, François, I think you have those viruses on the run. Don't get cocky, *mon ami*. The horde already is dragging away everything in your recipes directory. Shoot, François! Shoot! I know, they just keep on coming, but who said good system security meant you could sit back and watch? This is a war, and those little green guys aren't going to give you a moment's peace. Pay attention now. To the right now—ah, too late. A valiant effort, but now it is my turn.

What am I thinking? Our guests will be here any moment. Prepare the tables and make sure all the computers are up and ready to go. To work, François! Here they come. Welcome, *mes amis*, to *Chez Marcel*, where good food, good wine and Linux naturally go hand in hand. Please sit and make yourselves comfortable. We have some exciting software for you to sample today, and speaking of sampling...François, head down to the wine cellar. In the south wing, you'll find a case of 1992 Toscana Vin Santo from Italy. Bring it up for our guests *immédiatement*!

As you may be aware, *mes amis*, this month's issue is dedicated to security. We all know that Linux, by design and by its nature, is a far more secure operating system than a common legacy desktop OS. You know the one I am talking about. Odds are pretty good you've either seen it or (*gasp!*) even used it. Given that other operating system's problems with security (not to mention the extend, embrace and just plain take over philosophy behind it), it's not surprising that this theme tends to show up in a number of games. That's the feature of our menu today—call it the lighter side of system security.

The first game on today's menu is one we've served up before at *Chez Marcel*, but I just have to mention it again. I'm talking about Brian Wellington and Matias Duarte's *XBill* (see the on-line Resources section). Somehow, it seems like the perfect introduction to this menu. *XBill* probably is already installed on your computer. The command is simply `xbill`. If it is not installed, check your distribution CDs before you download a copy.

The idea is simple. It is your job to save the world's computer networks from an evil virus writer known as Bill, as he tries to load a virus (cleverly disguised as an operating system) onto otherwise healthy computers. Evil clones wander about the play area (Figure 1), stealing operating systems. To stop this from happening, click on the micro-clones using your mouse. Work fast!

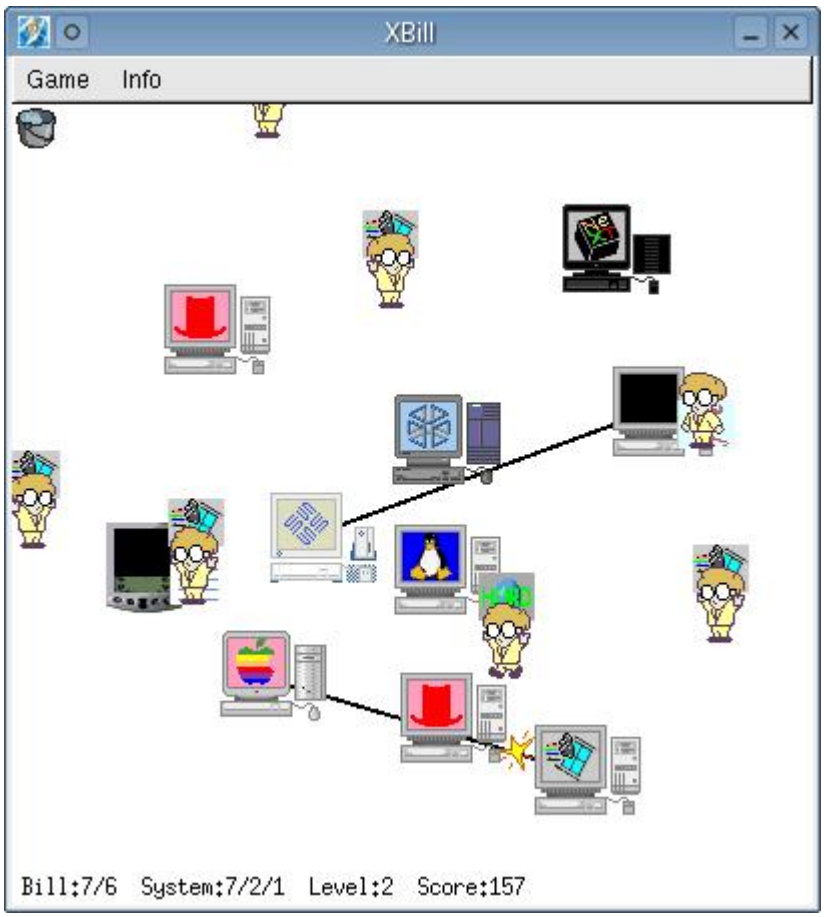


Figure 1. *XBill*: Can you stop Bill?

Once upon a time, there was a rather popular arcade game called *Defender*, which over time, spawned a huge number of imitators (or homages). One of these creations, Bill Kendrick's *Defendguin*, pits our hero Tux against evil aliens bent on enslaving, assimilating and mutating a planet of peaceful penguins into deadly penguinoids. These aliens, who look an awful lot like the guy named Bill from *XBill*, turn the helpless denizens into flying, killing machines. Once a penguin gets transformed into a penguinoid, there is no hope for the poor

beast—just blast it. Better yet, blast the alien intruders and get those points up (Figure 2).

Figure 2. Marcel loses yet another ship playing *Defendguin*.

The *Defendguin* site provides precompiled binaries for a number of platforms as well as the source code. To build *Defendguin*, you need the SDL development libraries. Should you choose to go the source route, it is a simple matter of extracting the bundle, typing `make` and then `sudo make install`.

To play the game, run the command `defendguin` and prepare yourself for a high-energy, fast-paced shoot-em-up. For full-screen action, add the -f option to the command.

Most modern offices running that other OS fight a never-ending battle against viruses. Trying to keep all those files and directories safe from dangerous intruders doesn't really sound like a game to those who fight the fight every day. But if you stop to think about it for a moment, you have to admit, it's the perfect inspiration for a game and that game is Stephen Sweeney's *Virus Killer*. Here's the premise: viruses attack your system through security holes in applications native to that other OS. Now, your files are in danger and the only thing that stands in the way is you.

Get your copy of *Virus Killer* at the Web site (see Resources). Binary and source RPMs as well as source are available from the site. *Virus Killer* is another SDL game. To compile it, you need the development libraries for SDL_image,

SDL_mixer and SDL_ttf. Make sure you have zziplib and its development library as well. Once the appropriate development libraries are in place, it is a pretty simple install:

```
tar -xzvf viruskiller-0.9-1.tar.gz
cd viruskiller-0.9-1
make
sudo make install
```

To play, simply type `viruskiller`. The program builds its list of files and directories based on your home directory. These then are the targets that the viruses in the game try to infect or otherwise destroy. Your job is to blast the little monsters before they can do any damage. As the Web site's FAQ points out, the viruses in the game do not actually destroy anything—it is only a game. If, however, you feel nervous seeing a little green monster drag away that document you've spent hours working on, try this version of the command: `viruskiller +safemode`.

This generates a file and directory list based on your system's /tmp directory and should help you feel better about those pesky viruses dragging away important documents.

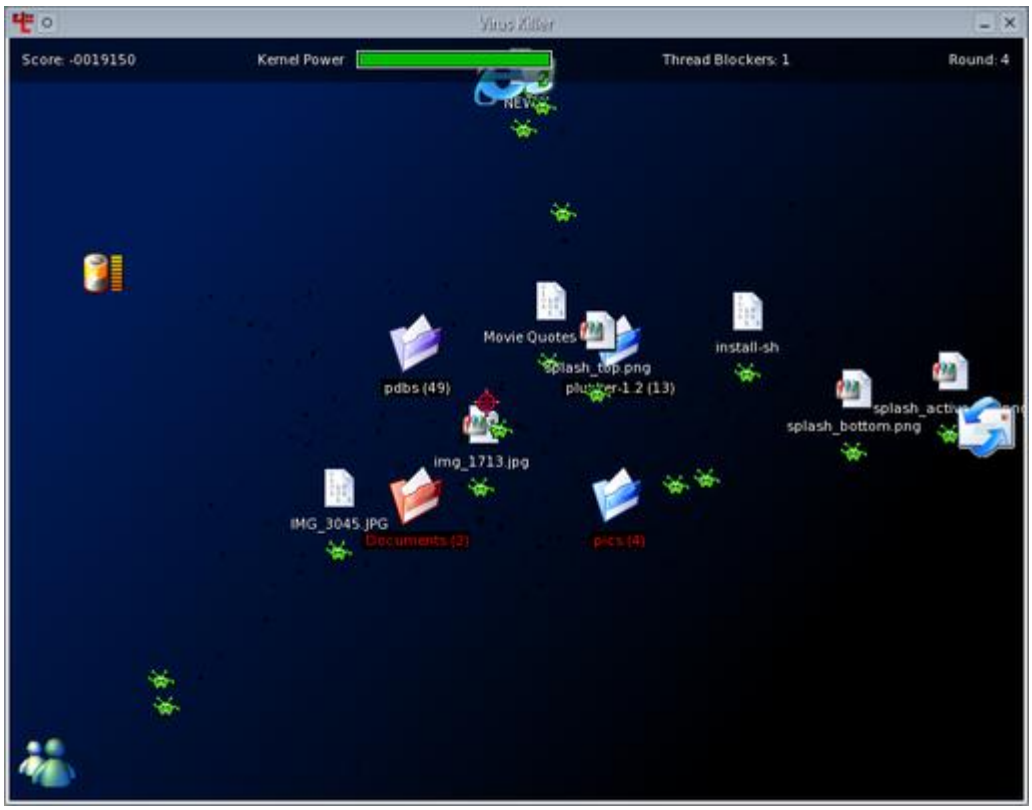


Figure 3. Shoot all the viruses before they destroy your files in *Virus Killer*.

The final item on today's menu is a little something called *Freedroid RPG*. Johannes Prix and Reinhard Prix's *Freedroid RPG* is a beautiful 3-D role-playing game with superb graphics, a cool soundtrack and sound effects, and a well-

developed world. Here's the backstory: sometime, in the not too distant future, a mega-corporation known as Megasoft effectively has taken over the galaxy. They managed to do this by using their vast corporate power to install trojan horses in every computer-equipped machine on the planet, including those of the government and police. As a result, all of humanity was enslaved. Due to some terrible programming error, however, the machines rebelled and took over, making things worse than they already were.

The only hope for mankind now is a cyborg version of Tux, a so-called lunarian. Equipped with high-tech armor, low-tech magic and a laser sword, our hero is ready to take on the machines and bring freedom to the galaxy. Along the way, you battle all sorts of villains and monsters, pick up items, money and weapons and meet up with all sorts of interesting characters. I even ran into a chef named Michelangelo whose favorite oven is down. He apparently needs dilithium crystals.

Can Tux help him repair it? The only way to find out is to play, and the only way to play is to get a copy of *Freedroid RPG* (see Resources). The download itself is a rather large file, around 60MB for the compressed tarball. This is, after all, an intensely graphical game. Source packages are available on the site, but binary packages are floating around for various distributions. For instance, I had no trouble finding an RPM package for Mandrake. Should you go the source route, building is an easy but slightly modified extract and build five-step:

```
tar -xjvf freedroidrpg-0.9.12.tar.bz2
cd freedroidrpg-0.9.12
./configure
make
su -c "make install"
```

As with the previous games, you need the SDL development libraries. If you are going to build the package with OpenGL support, you need those libraries as well. When the build and install is complete, you are ready to go. There are two ways to start up FreedroidRPG and these depend on whether you are running an accelerated OpenGL system or are stuck with plain old 2-D acceleration. The OpenGL method involves typing `freedroidRPG` (case matters here). To start in non-OpenGL mode, type this instead: `freedroidRPG -n`.

When the game first loads, you are asked to choose a single or multiplayer game and then to load a particular hero or start with a new one. If this is your first game, you obviously want to start with a new hero. Type in the name you want—François, for example. When you press Enter, the game loads and the adventure begins. If this is your first game, read the introductory information scrolling across the screen to learn the story behind the games, what the various keystrokes do and a lot of other useful information. When you have all

this down pat (it takes only a few seconds), click anywhere on the screen to begin.

Figure 4. All ready to head out on the *Freedroid RPG* adventure.

You can move around using your mouse in this 3-D world. Click ahead of your hero and he follows you. Click on objects to pick them up and right-click to activate your current skill or tool (your laser sword, for example). On the lower right-hand side of the screen are three circles. One brings up your hero's current statistics, another activates his inventory browser and another lets you select skills or tools (displayed in the lower left). At any time during the course of the game, pressing the Esc key brings up a menu from which you can modify various game-play options (graphics, sound, performance and so on). By default, game play is full screen but this is one of the settings you can change under the graphics options menu.

As you play, heed this friendly warning, *mes amis*. *Freedroid* (like all of the other games covered here today) is very addictive. You may find that time miraculously vanishes as you play. That's why I'm going to tell you about the save feature. If midway through a game, you need to pause to eat or sleep, press Esc to bring up the in-game menu, then select Save Game. You now can quit your in-process game safely and return to it later by choosing Load Existing Hero at the start of the game.

Mon Dieu, but it happens quickly. It seems we barely have started, and it already is closing time. I see that many of you are a little too wired from fighting the forces of poor security to head home right away. That's fine. My faithful

waiter, François, will pour you a final glass of wine. Perhaps in the time that you have left, you might take another opportunity to make the world safe from viruses, worms, evil corporations and other unsavory security risks. Until next time, *mes amis*, let us all drink to one another's health. *A votre santé Bon appétit!*

Resources for this article: /article/7685.

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. He is the author of the all new *Moving to the Linux Business Desktop* (ISBN 0-131-42192-1), his third book from Addison-Wesley. In real life, he is president of Salmar Consulting Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy, and folds a mean origami T-Rex.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Paranoid Penguin

Linux Filesystem Security, Part I

Mick Bauer

Issue #126, October 2004

Errors in setting permissions are the often-frustrating cause of many common Linux problems, so learn the fundamentals of permissions and take the first step to understanding Linux security.

For most of the Paranoid Penguin's illustrious four years with *Linux Journal*, I've tended to write tools-focused columns. I've described how to secure Sendmail, how to add SSL encryption to things by using Stunnel and how to get any number of other powerful security software tools configured and running.

Over the next couple of columns, however, I am going to address one of the most basic and important, yet often-overlooked aspects of Linux security; filesystem permissions. If used wisely, it will be harder for users and intruders to abuse their system privileges. If you set them carelessly, however, minor vulnerabilities can lead to major system compromises.

These articles should be especially useful to Linux newcomers who wonder what all the `drwxr-xr-x` gobbledygook in file listings means. But, even if you're an intermediate user—perhaps the kind who doesn't yet understand the precise ramifications of setuid and setgid—these articles, especially Part II, may have something for you too.

Prelude: Everything Is a File

Did you know that in UNIX and UNIX-like systems, basically everything is represented by files? Documents, pictures and even executable programs are easy to conceptualize as files on your hard disk. Although we think of a directory as a container of files, a directory actually is a file containing, you guessed it, a list of other files.

Similarly, the CD-ROM drive attached to your system seems tangible enough, but to your Linux kernel, it too is a file—the special device file /dev/cdrom. To send data from it or to write it to the CD-ROM drive, the kernel actually reads to and writes from this special file. Actually, on most systems, /dev/cdrom is a symbolic link to /dev/hdb or some other special file. And wouldn't you know it, a symbolic link is in turn nothing more than a file containing the location of another file.

Other special files, such as named pipes, act as input/output (I/O) conduits, allowing one process or program to pass data to another.

My point here is not to describe each and every type of file that exists in Linux or UNIX. It's to illustrate how nearly everything is represented by a file. Once you understand this, it's much easier to understand why filesystem security is such a big deal and how it works.

Commands and Man Pages

In this article, I focus on filesystem concepts rather than the precise syntax and usage of actual commands. But if you're a beginner, you may be wondering how to execute commands at all and where can you find syntax/usage help.

First, in all of my examples and example scenarios, I'm working in a terminal window. Microsoft Windows users can think of a terminal as like a DOS prompt or command window. A terminal window provides the most direct means of interacting with Linux, that is, by letting you enter all your commands manually rather than by triggering them with mouse clicks.

To start your own shell session from GNOME, click the Main Menu button and select System Tools→Terminal. In KDE, the terminal command is called konsole, and it has its own icon on the taskbar, a clamshell in front of a computer screen. Alternatively, you can start the Run Program dialog and type `konsole` at the prompt.

For fast help with practically any Linux command from within a terminal/shell session, you can type that command followed by the **--help** option. For example, if I can't remember all the command-line options for the ls command, which lists files and directories, I enter the command `ls --help`.

The **--help** option is quick, but it doesn't work for all commands. Even when it does work, its output can be quite terse. The best way to get command help is by using the man command. Man pages provide complete instructions on how to use most Linux commands and are present on practically all UNIX-like systems. To see the man page for the ls command, for example, type the

command `man ls`. Within the man page listing, press the spacebar to advance forward one page, the B key to go back one page and type `/somestring` to search the man page for *somestring*.

But, what if you don't know the name of the command you need? That's what apropos is for. For example, type `apropos list` to see a variety of commands that list things, and then pull up a man page for whichever of those commands seem to be what you need.

Users, Groups and Permissions

Actually, two things on a Linux system aren't represented by files, user accounts and group accounts, which we call users and groups for short. Various files contain information about a system's users and groups, but none of those files actually represents them. A user account represents someone or something capable of using files. This is to say, both human beings and system processes can use user accounts. For example, a user account named webmaster typically represents a human being who maintains Web sites. But the standard Linux user account lp is used by the line printer dæmon (lpd); the lpd program runs as the user lp. I explain later what it means for a program to run as one user vs. another.

A group account simply is a list of user accounts. Each user account is defined with a main group membership but may in fact belong to as many groups as needed. For example, the user maestro may have a main group membership in conductors and also belong to pianists.

A user's main group membership is specified in the user account's entry in /etc/password. You can add that user to additional groups by editing /etc/group and adding the user name to the end of the entry for each group to which the user should belong. Alternatively, you could use the usermod command; see the usermod(8) man page for more information.

Listing 1 shows maestro's entry in the file /etc/password, and Listing 2 shows part of the corresponding /etc/group file.

Listing 1. An /etc/password Entry for the User maestro

```
maestro:x:200:100:Maestro Edward Hizzersands:/home/maestro:/bin/bash
```

Listing 2. Two /etc/group Entries

```
conductors:x:100:
pianists:x:102:maestro,volodyia
```

In Listing 1, we see that the first field contains the name of the user account, maestro. The second field (x) is a placeholder for maestro's password, which actually is stored in /etc/shadow. The third field shows maestro's numeric user ID, or uid; in this case it's 200. The fourth field shows the numeric group ID, or gid—in this case it's 100—of maestro's main group membership. The remaining fields specify a comment, maestro's home directory and maestro's default login shell.

In Listing 2, from /etc/group, each line simply contains a group name, a group password (usually unused—x is a placeholder), numeric group ID (gid) and a comma-delimited list of users with secondary memberships in the group. Thus, we see that the group conductors has a gid of 100, which corresponds to the gid specified as maestro's main group in Listing 1. We also see that the group pianists includes the user maestro, plus another named volodyia, as a secondary member.

The simplest way to modify /etc/password and /etc/group in order to create, modify and delete user accounts is by using the commands useradd, usermod and userdel, respectively. I'd rather concentrate here on concepts than command syntax, so suffice it to say that all three of these commands can be used to set and modify group memberships and all three commands are well documented in their respective man pages. To see a quick usage summary, you also can type the command followed by **--help**, for example, `useradd --help`.

Simple File Permissions

Each file has two owners, a user and a group, each with its own set of permissions that specify what the user or group may do with the file—read it, write to it and execute it. A third set of permissions pertains to what others, user accounts that don't own the file or belong to the group that owns it, can do with the file. Listing 3 shows a long file listing for the file /home/maestro/baton_dealers.txt.

Listing 3: File Listing Showing Permissions

```
-rw-rw-r--  1  maestro  conductors  35414  Mar 25 01:38 baton_dealers.txt
```

Permissions are listed in the order of user permissions, group permissions and other permissions. For the file shown in Listing 3, its user owner (maestro) can read and write the file (rw-); its group owner (conductors) also can read and write the file (rw-), but other users can only read the file. Permissions are a little more complicated, however. Users classified as other, in terms of permissions on a particular file, can delete any file in a directory to which they have write

permissions. In other words, users with read-only permission on a file cannot edit the file but can delete it if they have write permission on the file's directory.

There's a third permission besides read and write: execute, which is denoted by x when set. If maestro writes a shell script named punish_bassoonists.sh and if he sets its permissions to **-rwxrw-r--**, he then can execute this script by entering the name of the script at the command line. If, however, he forgets to set the execute permission, he is not able to run the script, even though he owns it.

Permissions and root

In practical terms, file permissions simply do not apply to the root user; root can do anything to any file, at any time. This is why it's so important never to log on as root or use the su command to become root, except when absolutely necessary. When you're root, file permissions do not protect you from your own mistakes.

This is not to say that all programs entirely disregard file permissions when you're root. If, for example, root tries to alter a read-only file using the vim editor, root must use the :w! command (force write). The normal ZZ or :w commands return an error in this case. However, many other commands have no such sanity-check feature.

Permissions usually are set with the chmod command, short for change mode. Continuing with our example, suppose maestro has second thoughts about allowing other members of the conductors group to read his list of baton dealers. He could remove the group read/write permissions using the commands shown in Listing 4.

Listing 4. Changing a File's Permissions with chmod

```
bash-$ ls -l baton_dealers.txt

-rw-rw-r--  1  maestro  conductors  35414  Mar 25 01:38 baton_dealers.txt

bash-$ chmod go-rw baton_dealers.txt
bash-$ ls -l baton_dealers.txt

-rw-------  1  maestro  conductors  35414  Mar 25 01:38 baton_dealers.txt
```

In Listing 4's sample chmod command (`chmod go-rw`), `go` tells chmod to change the group permissions and other permissions; `-rw` says to remove read and write permissions for those two categories of permissions, group and other. Thus, a chmod command has three parts: a permission category, some combination of u, g and o or a for all; an operator, - to remove, + to add; and a list of permissions to add or remove, usually r, w or x.

Directory Permissions

We now know how to change basic permissions on regular files, but what about directories? Directory permissions work slightly differently from permissions on regular files. Read and write are similar; for directories these permissions translate to list the directory's contents and create or delete files within the directory, respectively.

Execute is a little less intuitive on directories, however. Here, execute translates to use anything within or change working directory to this directory. That is, if a user or group has execute permissions on a given directory, the user or group can list that directory's contents, read that directory's files (assuming those individual files' own permissions include this) and change the working directory to that directory with the command cd. If a user or group does not have execute permission on a given directory, the user or group is unable to list or read anything in it, regardless of the permissions set on the things inside. If you lack execute permission on a directory but do have read permission and you try to list its contents with ls, you receive an error message that, in fact, lists the directory's contents. But this doesn't work if you have neither read nor execute permissions on the directory.

Suppose our example system has a user named biff who belongs to the group drummers. Also suppose that his home directory contains a directory called extreme_casseroles that he wishes to share with his fellow percussionists. Listing 5 shows how biff might set that directory's permissions.

Listing 5. A Group-Readable Directory

```
bash-$ chmod g+rx extreme_casseroles
bash-$ ls -l extreme_casseroles

drwxr-x---  8  biff  drummers  288  Mar 25 01:38 extreme_casseroles
```

Per Listing 5, only biff has the ability to create, change or delete files inside extreme_casseroles. Other members of the group drummers can list its contents and cd to it. Everyone else on the system, however, is blocked from listing, reading, cd-ing or doing anything else with the directory.

Conclusion (for Now)

Those are the most basic concepts and practical uses of Linux filesystem security. In Part II, we'll go further in depth and discuss (among other things) setuid, setgid and numeric permission modes. Until then, be safe!

Mick Bauer, CISSP, is *Linux Journal*'s security editor and an IS security consultant in Minneapolis, Minnesota. He's the author of *Building Secure Servers With Linux* (O'Reilly & Associates, 2002).

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Linux for Suits

Bridging the Gap

Doc Searls

Issue #126, October 2004

Doc visits an old-fashioned, top-down, IT organization where “open” is a dirty word and the number one priority is avoiding uncertainty.

If you're one of the growing number of Linux advocates working in corporate IT, chances are you're involved in a struggle. Your side, the Linux/open-source side, is a grassroots movement, usually driven from the bottom up, not from the top down. Meanwhile, IT management remains a top-down culture that prefers “solutions” in the form of mandated strategy, not ad hoc problem solving.

To understand what we're up against, it helps to visit the top-down side where it lives. That's what I did on June 27–30, 2004, in Boston, at the 32nd Annual Conference and Annual General Meeting of the Membership of the Information Systems Audit and Control Association, the ISACA.

For all of its 32 years, ISACA has been obsessed with governance, control, accountability, strategy, planning, risk reduction, business continuity and so on. Here's a sample of the educational sessions:

- “'Who Has Access to What': Controlling User Access and Segregation of Duties with Identity Management”
- “Enterprise Risk Management and Basel II: The Mission Critical Role of IT”
- “Incident Response Forensics Investigation”
- “Use of Encryption to Enforce Regulatory Compliance”
- “Successful Software Asset Management”

Not surprisingly, open source wasn't on the agenda—until this year, when I was invited to come and teach a class on “The Realities of Open Source”.

Before and after I gave my talk, I sat in on a variety of other sessions. In every case I carried the only laptop in the room, other than the ones used by speakers to give Microsoft PowerPoint presentations. When I pulled it out to take notes, or to work on my own talk, others looked at me like I was carrying a loaded weapon. Was I about to commit an act of wireless hacking, perhaps?

That would have been cool, of course, but not with this crowd. Open source, to them, smacks of the unknown. The very notion of “open” sounds like a security breach waiting to happen—a risk best “managed” by avoiding it in the first place.

So giving the talk was a fun challenge. I had addressed newbies before, but never newbies who also happened to work in IT Governance. I felt like I was explaining free enterprise to the Politburo in the old USSR. The cultural distance verged on the absolute.

I opened with the two questions I already knew were on everybody's mind:

1. How do you design an audit to account for open source?
2. For that matter, how do you govern, control and assure it?

I answered by saying “We're not in Kansas anymore” and proceeded to explain how many of the civilized graces we take for granted, from e-mail and Web servers to Amazon and eBay, owed credit to open-source developers, values and effects—all of which were as wild and uncontrollable as the next storm or earthquake.

That said, I went on to explain that open source was also about *resourceful individuals and groups doing what needs to be done*, that open source is an example of *what happens when the demand side supplies itself* and that it can be understood as the *DIY sector of the Information Systems construction and maintenance business*.

I also said none of this was especially visible when you looked down at it from the parapets of Fort Business, demanding that it comply with approved procedures before Management dropped the bridge across the moat.

I showed the sizes of various open-source conversations on the Web, which equaled or even exceeded those concerning Microsoft and other familiar names. I showed how much Linux and open-source code probably was already running in attendees' companies, whether they knew it or not.

And, I talked about common interests—for example, the use value (rather than the sale value) of software, and also low cost, availability, quality, integrity,

efficiency, effectiveness, continuity, robust infrastructure, leveraged physical assets, problem solving and other virtues copied from the slides of other speakers.

In the Q&A we talked about SCO FUD and other predictable issues. One attendee demanded an explanation for why “they” (meaning open-source advocates) had “such terrible manners”. Later on a boat tour of Boston Harbor, she came up to me, waved a cigarette in my face, and insisted that open-source people “have a good case, but they simply must change their behavior”. But she couldn't give me an example of what she meant.

After my talk, several attendees told me they were, in fact, open-source users and advocates inside their organizations (one was Monsanto), but that the prospects for open source in large conservative IT organizations was poor in the short term, even if they were good in the long term.

The main issue was auditability. One attendee told me that open-source stuff is, indeed, auditable (for example, the source code itself can be audited), but that this fact was still far from obvious.

It was clear, however, that for most attendees, open source simply wasn't on their radar. “We don't have any, and we don't plan to have any”, was the basic response. Another sign: the room was only half-full. The session after mine was on “Wireless Hacking Exposed”. It was standing-room only.

In my July 9, 2004, SuitWatch newsletter, I wrote about the cultural divide I witnessed at the conference and its near-absolute contrast from Supernova, the conference where I spoke a couple of days before ISACA. The theme for Supernova was decentralization. If anything, ISACA's theme was the polar opposite.

Immediately an e-mail came from Glen Campbell, an IT consultant and veteran of both cultures. “The largest driving factor in most Fortune 2000 IT shops is risk mitigation”, he said. “Likewise, the whole concept of DIY-IT is a complete anachronism to most IT shops. They did DIY-IT in the '60s, '70s and '80s, and discovered that, not only is it less expensive to outsource their internal systems (to SAP, or Siebel, or EDS or IBM), it also—tada! —reduces their risk in that there's another large corporation out there they can sue if something goes wrong.”

Customers requiring “six nines” of reliability have been getting it for years from those same providers, or from HP, Sun or Novell. “That's why Sun can charge 10–50× for a server that runs Apache (which, by the way, is one of the very few

open-source packages that large companies have come to trust)", Campbell adds.

The regulatory environment has also elevated corporate IT paranoia.

Last year I attended a Harvard Business conference in New York where a procession of expensive consultants from Deloitte and other companies described the complex new reporting and accounting procedures required by the recently passed Sarbanes-Oxley Act, which enforces, post-Enron, a much higher degree of corporate—and personal—accountability. Regardless of whatever good it intended to do, it also was clear that Sarbanes-Oxley promised to (borrowing Scott McNealy's immortal words) "darken the skies with consultants" and expand corporate bureaucracies—IT Governance, for example. It was clear at ISACA that Sarbanes-Oxley was a godsend for members. Suddenly they found themselves standing exactly in the direction top corporate brass was forced by the new law to turn to for help.

So, the question becomes, where does open source fit in a Sarbanes-Oxley context? Here's Glen Campbell again:

> I have a number of friends here in Silicon Valley who are on the "high-tech" side of the world. They simply cannot understand why, when you encounter an IT problem, a company doesn't grab Perl or Python and write a script that solves it. The IT guys, on the other hand, look at this with horror; how in the world could this hold up to a Sarbanes-Oxley investigation, which requires the CEO to sign off personally on the sources and validity of the corporate data? How in the world is the CEO going to understand a Perl script? The CEO will happily pay $100,000 to avoid going to jail because of this.

And his isn't the only bad news. Another SuitWatch response included this:

> ...my organization's overlords, some of whom were probably at ISACA and most assuredly none of whom were at Supernova, have issued a blanket edict forbidding the use of Wi-Fi anywhere inside the entire (organization) and forbidding the use of Wi-Fi with any (organization) equipment anywhere at any time. Needless to say, the (organization) CIO is firmly anti-open source and appears to be a vassal of Microsoft.

Yet Linux continues to grow in the enterprise. So does the whole LAMP suite. I'm writing this column in London, at the Identity Management Summit, where I just finished talking with Conn Crawford, who runs IT for the City of Sunderland, on the northeast coast of England, and with Nick Somper, Business Development Manager of Open Systems Management, Ltd. Both had Linux

stories to tell. Crawford said his city just bought a Red Hat server to issue certificates for the city's PKI system. Somper's company does “adaptive systems management” for distributed and mixed environments that include UNIX, Linux and Windows systems. He sees Linux as a large and growing presence in IT.

Crawford and Somper both live in the market space where Linux has grown past the grassroots stage. It's part of the IT ecosystem in that space. Now the challenge is to make Linux more than safe—in perception as well as reality—for the large IT shops where risk aversion is a way of life. But before we do that, we need to understand the differences involved.

Adam Hertz, CTO of Ofoto, has a shop where DIY-IT and open source are ways of life. The company's whole business runs on systems developed almost entirely internally, on Linux and other open-source components. It also grows at an extremely fast rate, adding up to two terabytes of storage every day. His whole business wouldn't be possible without open source, Linux and DIY-IT. From that perspective, here's what he sees in the relatively staid environments where most ISACA members live:

> Two themes: BigCoIT is all about standardization and isolationism.
>
> Standardization, so the story goes, reduces risks and costs. It certainly reduces complexity, but it can take a huge toll on flexibility and responsiveness.
>
> Standardization often involves using one multipurpose tool or platform to accomplish a lot of different purposes. This often involves customization, which is done by in-house experts or professional services firms—for example, Siebel, Lotus Notes and so on.
>
> DIY shops tend to be cynical, or even downright frightened, of systems like that, because they're so inflexible and unhackable.
>
> Another form of standardization is what people are allowed to have on their desktop PCs. In a lot of big shops, everyone has the same disk image, with all applications pre-installed. There's a huge suspicion of anything that comes from the outside world, especially open source. It's regarded as flaky, virus-laden, unscalable and so on. This produces isolationism, which means major barriers to just trying something.
>
> In more open environments, there's a permeable membrane between the corporate IT environment and the Net. People tend to get new tools from the Net, usually open source, and just give 'em a spin. Culturally, this keeps the organization open to innovation and new approaches. It builds bonds

> between the employees and the development community at large.
>
> Standardized, isolationist shops miss out on all of this. They maintain control, but inevitably they fall behind.

If we want to sell Linux upward into risk-averse enterprises, it would help to follow the guidance, as well as the example, of J.P. Rangaswami, CIO of Dresdner Kleinwort Wasserstein, the billion-dollar-plus investment bank. When I asked J.P. to help make sense of the challenge here, he said:

> Too many shops are currently executing uncertainty management rather than risk management. And because they don't understand the risk, they do the right thing and sit on their hands. Once you understand a risk you are able to price it and make a rational decision. So in my opinion, shops differ in their ability to price risk, and the default position is risk aversion, which may be far more expensive even in the short run than an active risk appetite. Witness the colossal sums spent on Y2000 just because people could not understand or price the risk.
>
> DIY-IT is, when based on open source, defensible as a low-risk position. More people inspect the code. There are more people using the code. There is more flexing of the code in differing circumstances. The issue is not of risk but of comprehension.

Last December Risk Waters named J.P. its CIO of the Year.

Resources for this article: /article/7691.

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

LINUX JOURNAL

EOF

Dear Laptop Vendor

Lincoln D. Durey

Issue #126, October 2004

Hardware vendors, please sell me a high-quality laptop without a pre-installed OS. We'll both make money.

I'm a collector, but I have one collection I alternately love and hate. I have several thousand “Designed for Microsoft Windows” stickers. I love them because each one marks a laptop that's been freed from the Microsoft matrix. I hate them because they remind me that any laptop you'd actually want to own for three to four years is born into slavery by its maker.

EmperorLinux has been installing, configuring, testing and supporting Linux on brand-name laptops and notebooks since 1999. This is not easy work. Getting Linux running well, with full support for all the hardware on a freshly minted laptop, presents many significant technical hurdles. Any Linux laptop worth its salt requires quite a raft of distribution-specific and system hardware-specific configuration file modifications, many of which are beyond current Linux automated configuration tools. Of course, no modern laptop will have full hardware support without a very heavily patched kernel. Customers—rightfully so—want solid support for IEEE-1394 FireWire, ACPI (with suspend), Wi-Fi (both b and g), external VGA and everything Windows users get from a laptop. No stock Linux distribution kernel can do this, and the Linux vendors all disclaim laptop support. Hardware vendors IBM and Dell even have dipped their toes into the Linux laptop pool occasionally and pulled out for reasons of support (lots) and demand (small).

Consumers should know that no name-brand vendor will sell them a laptop without a preloaded operating system. They can be certain that the OS will be supplied by a much-maligned monopoly. The problem, of course, is in the nature of the Faustian bargain into which all major hardware vendors have entered in order to get where they are today. Every time the big vendors send

their representatives to us, they ask what they can do for us. I tell them their hardware design is excellent and that my customers appreciate the long-term hardware warranties. I ask only one thing they don't provide, and I am vexed by the unchanging answer. Even a big vendor's authorized reseller or strategic partner cannot get this mythical “bare” laptop.

Our customers know, I know and the big vendors probably know these things hold true: 1) laptops are a small percentage of the total personal computing market; 2) Linux, though big on the server, is a small part of the desktop OS market; 3) “Linux” doesn't mean one thing, there are at least six viable, popular Linux distributions, each having two to three versions in common usage; and 4) taken together, those imply that there can't possibly be a decent market.

We know that big business is about profit, and that is fine. Even though I'm a PhD geek who likes laptops, at the end of the day, I've got a whole mess of hungry little Georgia Tech graduates to feed. We know that neither the stability of Linux nor its high ideals really factor in your decision. If your projections don't yield the right ROI numbers, the product idea behind those projections won't get off the drawing board.

There is a way out of this, which just happens to be a win-win situation for everyone. Sell “bare” laptops. These must be exactly the same systems you purvey with Windows, only with bare disks (simply add a “b” to the end of your model stock codes). Clearly mark these units as having full hardware warranties but absolutely no software support for any OS (the Linux users aren't going to call you anyway). These systems need not be massively cheaper than your Windows offerings either. In fact, we all know the “Microsoft Tax” is really only a moral affront and that the dollar cost is well under $100. Don't try to push a lower-end system on us; we want the full-feature selection and per-specification configurability you otherwise would offer. Plenty of Linux users are out there who want a $3,000 laptop.

Do all this, and here is what you will get: 1) a massive flow of Linux user goodwill—the moral aspect really is important to us; 2) your foot quickly in the door of a very rapidly growing market; 3) the ability to sidestep the whole “Which Linux distribution?” question, and its support issues, by disguising it as “End-User Choice”; and 4) all your high-margin laptop hardware profit up front, with zero software support costs later. In short, you will get a new business line with a very short time-to-market, high ROI and low risk.

Let's be frank, Mr Big Vendor, you are in the hardware business and have no resources nor desire to get into the OS software support business with individual customers. That is what my company does. Please take our money and laugh all the way to the bank. We'll do all the hard work.

Lincoln Durey is founder and president of EmperorLinux (www.EmperorLinux.com). He was introduced to Linux in 1994 while working on his PhD in Electrical Engineering. He has been putting Linux on laptops since 1999.

Archive Index Issue Table of Contents

Advanced search

IBM's IntelliStation A Pro

Chris DiBona

Issue #126, October 2004

IBM has created a capable Red Hat Linux ES-based workstation featuring dual AMD Opteron processors with gobs of room for expansion.

Product Information.

- Vendor: IBM
- URL: www.ibm.com/intellistation/pro
- Price: $7,066 US

The Good.

- Fast as all get-out.
- Very expandable.
- Great video.

The Bad.

- Memory “hole”.
- Loud as heck.
- Noisy audio.
- Spendy.

Although the shipping now of a high-end Linux workstation from a major computer manufacturer is not the news it was in the late 1990s, the arrival of the A Pro workstation is nonetheless important and not only for its pedigree. With it, IBM has created a capable Red Hat Linux ES-based workstation featuring dual AMD Opteron processors with gobs of room for expansion.

The A Pro provided for review has the following features:

- Two AMD Opteron 248 series processors running at 2.2GHz each.
- 4GB, um, 3GB of memory out of a possible 16GBs (more on memory below)
- One 36GB SCSI hard drive.
- An NVIDIA Quadro FX 1100 GPU.
- Mouse/keyboard and other miscellaneous bits.

As shipped, the A Pro costs $7,066, according to the IBM store on May 25th, 2004. Options are available.

When I worked for Linux hardware company VA Research, we always were amazed at—no matter how much integration work we'd put into customizing a distribution—how many of our customers would reload the machine with their favorite flavor of Linux as soon as it hit their loading docks. I'm not certain this situation has changed, but I've chosen to review the machine as delivered.

The review machine did not ship with recovery media, but the contact from IBM has assured me that the machine does ship with recovery CD-ROMs, so that's good.

The Hardware

The first thing you notice after slapping the Power button is the sound. Is it quiet? Well, for a dual Opteron, the noise level is what one would expect. IBM has done some nice work to cut down on the noise, such as adding rubber grommets on the fan mounts and providing some solid thick steel in the construction of the workstation's case. It is not whisper-quiet by any means, and I think that will limit the A Pro's use to the professional who needs this much capability. As I noted before, volume is a relative thing, and I think this system probably is quieter than it should be, considering the expansion capabilities presented by the A Pro. Also, this thing weighs a ton.

One nit about the hardware: it should be noted that it didn't recognize my monitor without tweaking. Not recognizing my monitor, an NEC Multisync FP1350x, was pretty odd, but no worries, redhat-config-xfree86 worked well enough to get X up and running.

I did not have an opportunity to span the screen to two monitors; as shipped, the NVIDIA driver does support such a configuration. The NVIDIA driver is proprietary software but commonly is used because it is considered to be faster than the open-source driver and supports more of the card's features.

An odd thing was some obnoxious noise (tick tick), over the audio subsystem (spurt tick), and it occurred whether you (tick) had your phones plugged in the front or back jacks. It really was annoying, and considering the system noise this thing puts out, the additional noise might qualify as a quality-of-life issue.

IBM engineers thoughtfully provide extra screws attached to the strut spanning the card cage in front of the motherboard, but inexplicably, they don't tie down or otherwise tuck away any of the internal cables, which can impede airflow and otherwise disrupt the machine's thermal performance. You could write this off to the pre-production nature of review machines, but it also can indicate a less than careful hand at manufacturing. Another nit is the quality of the mouse and keyboard. The keyboard is your standard generic keyboard, but the mouse is awful and made me want to smash it with a hammer.

The machine they sent came with 4GB of memory. The BIOS, however, has a hole of 1GB, making the fourth gigabyte disappear. So, effectively, if you buy a machine with more than 3GB of memory—something 64-bit computer users want, mind you—you are paying a tax of one gigabyte.

I wouldn't recommend buying an A Pro until IBM fixes this memory loss situation. This is doubly vexing because at the time of this writing, IBM was offering machines with up to 16GB on its Web site, with no note that you would be losing out expensively. [*IBM has since updated its Web site with information on the memory hole issue. See sidebar. —Ed.*]

The Software

The A Pro appears to be aimed solidly at a graphic workstation market. As such, I tested it using some GIS exploration software I have experience with, so I could get a feel for how well IBM has succeeded at providing a machine for that market.

The machine ships with a pretty standard Red Hat Enterprise Server load. As mentioned above, there were a few hiccups during load, mostly connected to video configuration and port selection. Past that, it doesn't look like IBM did much at all to the distribution, short of loading the NVIDIA driver as noted above. Although I'd like to see more “welcome to your A Pro Super Machine, here's what you've got”, a stock distribution is less likely to be overwritten immediately, so IBM's decision is understandable.

Partitioning was done competently in my opinion, with /var suitably reined in. When you are in the Linux hardware business you quickly realize there is no one true partitioning scheme. I imagine you would want to tell IBM your preferred layout if you were to buy in quantity, though. Like many Linux users, I have a set of things I do to any base load—change Mozilla's search to Google,

set up the monitor to reflect my ideas about color depth and resolution—and the system didn't gag on anything I did to it in that vein.

Application Support

The A Pro comes with the regular array of apps, including Mozilla, OpenOffice.org and everything else that ships with Red Hat ES. IBM chose to load all the apps, which is a valid decision for a desktop machine. Again, when buying, you might want to specify the parameters of your load.

Java is the only major language not supported out of the box, unless you consider Pascal major. This struck me as odd considering IBM's dedication to the language.

Tux Racer

Games are a good way to test the OpenGL subsystem, and they also are fun. *Tux Racer* ran fine—no graphic glitching. The sound was not turned on out of the box, though, so I set that up using the standard Red Hat sound card setup program to good result. It struck me as odd that IBM hadn't done this as part of the load.

TerraVision Test

Because I wanted a program that could test the graphics, processors and memory handling of this machine, I chose the geographical exploration application TerraVision, which allows you to do 3-D flythroughs of GIS height map data with textures created from satellite data. I loaded four data sets into the system—the Palo Alto resolution set, global set, Lawrence Livermore and San Francisco—all at various resolutions, spanning 1km to 3 meters. This represents a total of about 4GBs of data, and although the program is good with swapping the data into the program's memory, it is quite consumptive. The A Pro performed quite well—no real glitching of the data set and the flythrough was smooth. You can download TerraVision and a number of data sets off the SRI Web site (www.ai.sri.com/TerraVision).

Price and Conclusion

When comparing the A Pro against two white box vendors, the IBM machine seems to run about $1,800 US more. Is it worth it? Perhaps, as IBM's casework and layout generally is excellent, but I have to admit I wasn't overwhelmed with the machine to the tune of $1,800, mostly due to the noise profile. I have to say, from a sound perspective, it was a relief to shut off the A Pro, which is never a good thing. The Opteron is a cool-running 64-bit chip, so it shouldn't take this many decibels to cool it.

The case is designed to hold and power a vast number of drives, so it has cooling to spare, but there is no clear way to turn down the fans and make them less intrusive. The scope of the machine should be taken into account as the valid excuse for the noise level, however. If you need the beefiest, most solid workstation on the market, then the IBM A Pro is what you'll be buying, and I think you'd be both happy and well served by it.

IBM Responds to the “Memory Hole”

The IBM IntelliStation A Pro uses an x86 industry-standard method to assign memory space for drivers of installed devices, which indicates that less memory is available to the operating system than the actual “physical” memory installed in the system. The memory reported by the operating system does not account for space reserved for the drivers that must reside in memory for devices such as the network controller, SCSI, IEEE 1394 and graphics cards.

The A Pro is packed with integrated technologies to provide our high-end customers with the scalability and functionality they demand. The industry-standard method implemented in the A Pro has an architectural limitation, because it is an x86-based system that must support both 32- and 64-bit operating systems. Therefore, the system BIOS must allocate memory for these device drivers within the first 4GB of memory in order to support the 32-bit operating systems.

As the first worldwide tier-one vendor to ship x86 64-bit capable workstations, IBM is leading the design of new solutions to take advantage of the full capabilities of the IntelliStation A Pro. We are in the process of working with our hardware and software vendors to design an open solution to eliminate the memory restriction in systems running a 64-bit operating system in future releases of the A Pro.

Chris DiBona is the cofounder of Damage Studios and works on the new massively multiplayer on-line role-playing game *Rekonstruction*. Chris is an internationally known open-source advocate. He writes for the *LJ* Web site and the book he co-edited, *Open Sources*, won the *Linux Journal* book-of-the-year award in 1999. A sequel currently is in the works. His personal Web site can be found at dibona.com.

Archive Index Issue Table of Contents

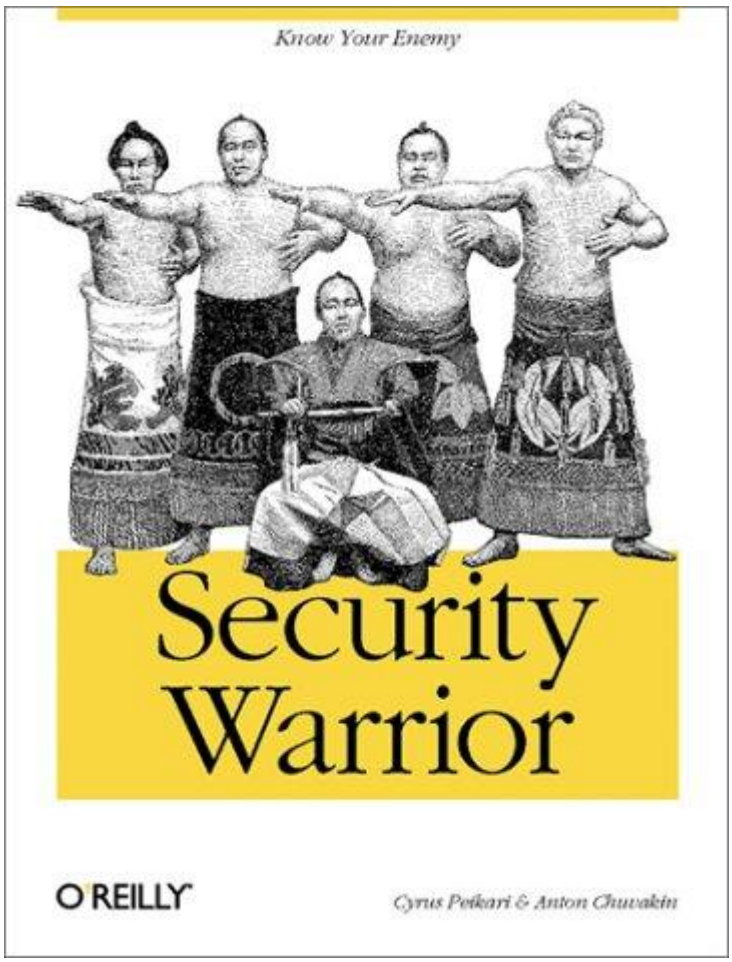
Advanced search

Security Warrior by Cyrus Peikari and Anton Chuvakin

Dan York

Issue #126, October 2004

This book lives up to its claim.



O'Reilly & Associates, 2004

ISBN: 0-596-00545-8

$44.95 US

Security Warrior is, indeed, a dangerous book. Not so much for the specific tools and techniques it presents, as all of them can be found on the Internet, but because the book collects all this information in one convenient, easy-to-read volume. With a subtitle of “Know Your Enemy”, this book provides a powerful compilation of attacks against software, networks and individual systems.

Given that hundreds of security books are out there, I was a bit skeptical this one would live up to its claim of being so different. However, as soon as you enter the first section, “Software Cracking”, you know you are in for a different ride. After a quick refresher on assembly language, this section covers how to reverse engineer software in Windows, Linux and Windows CE, with the focus on how to crack malware such as viruses or spyware. I personally found this section a bit slow-going, but I did learn a good bit from it. I especially found the text on overflow attacks quite relevant, given the large number of such attacks around today.

For me, the book really hit its stride in the second section, “Network Stalking”. After a brief review of basic TCP/IP attacks and tools, the text dives into active and passive reconnaissance, OS fingerprinting and hiding an attack. Chapter 7, on social engineering, seems a bit out of place in this section, but it is an interesting read nonetheless. In later sections, I enjoyed the well-written chapters on hardening UNIX/Linux systems and UNIX/Linux attacks, which include information about breaking out of chroot jails that I hadn't seen in other security books.

My only minor complaint about the book is the editing is a little uneven. Most sections are well done, but in a few cases there are references to topics that “would be covered later” but never are. In another case, I felt there was unnecessary duplication of information. Overall, I found this book to be a strong text with a refreshingly different spin on computer/network security. If you are responsible for system or network security, *Security Warrior* is definitely worth reading.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

From the Editor

Security

Don Marti

Issue #126, October 2004

The security message is closing Linux migration deals. Now it's up to us to deliver the secure environment people want.

You can talk about cost savings, performance and flexibility all you want, but the advantage driving more and more companies toward Linux is security. Just look how much time the big cheeses in the proprietary OS business spend telling the media about their catch-up plans. Thanks to some bad mistakes in the design of one vendor's browser and mail client, CIOs are asking vendors for Linux answers faster than the vendors were expecting.

Some OSes are born ubiquitous, others attain ubiquity and Linux is having ubiquity thrust upon it. Customer pull is nothing new to the Linux vendors, and they'll cope. And for you, the Linux professional, it's opening night at the big show. Everyone bought a ticket to see the amazing singing, dancing, secure operating system. They're waiting for the curtain to go up, and you're the stage manager.

Don't panic. Security depends more on policies and attention to detail than on any program or product. And you have a secret weapon. As you move more systems to Linux, you can start enforcing more secure policies and conceal the changes in the smoke and mirrors of the OS migration. If anyone points out that you could relax security to the way you had it in your old OS, you can say “that's the way it's normally done under Linux.” Yes, Linux will get some of the credit for your good decisions, but you'll get credit for putting in Linux.

Everyone will tell you to run Nmap to keep track of open ports and get an early warning of unnecessary or misconfigured software, but when you're keeping track of thousands of systems, that's a lot of data to watch. Log your Nmap data to an SQL database with Hasnain Atique's article on page 56.

Makan Pourzandi and Axelle Apvrille are bringing security to the Linux cluster environment (page 64). If you're sharing a cluster among multiple project teams, have a look.

SELinux is one of the most promising developments in Linux security, and it's worth keeping an eye on. No more will an attacker be able to “get root” on a whole system by compromising one dæmon. I'm planning to use SELinux at first for simple bastion hosts such as name servers, then add it to other systems as the administration tools get better. SELinux is complicated, though, so watch *Linux Journal* for more articles about it. James Morris explains SELinux and filesystems on page 22.

Finally, we normally don't bother with making fun of proprietary operating systems, because we're just quietly replacing them and interoperating with them where they're still in use. But Marcel Gagné got a little too annoyed by the latest batch of worms targeting other OSes that clobbered his network, so he blew off a little steam with some games on page 30. Have fun, keep your systems secure and enjoy the issue.

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

Archive Index Issue Table of Contents

Advanced search

Letters

Readers sound off.

Wireless Questions

I have been reading *LJ* for a few years now, and I am amazed at the transformation of the articles and content. I especially like Robin Rowe's articles about technology in the movie industry. He tends to have a lot of inside info about DreamWorks and ILM that I find interesting. Please keep these articles coming. Marcel gives some great tips for newbies and casual experimenters, which I appreciate, and I think the French theme is orginal and entertaining. There was an article about GNU Radio in the April 2004 issue. The article glazes over the issue of how the input stream is obtained from the A/D (sound card or PC Card) and piped to the processing application. I have an interest in this software for use in my master's thesis. Could this be explained?

One more thing—what is an inexpensive PCMCIA wireless card that is compatible with Linux?

—
Michael Wallace

The September 2004 article by Eric Blossom featured a step-by-step FM listening project that should help with your question. Hope you enjoyed it. All of the common wireless cards are compatible—check your distribution's hardware list. The least expensive one we use is an Actiontec HWC01170-01. —Ed.

Check Out gdesklets

I am pretty new to the *Linux Journal* world, so forgive me if you have already covered this topic, but I recently read an article about SuperKaramba [*LJ*, May 2004] and just wanted to suggest writing about gdesklets as well. In my experience, they are much harder to install and actually get working than SuperKaramba, and I think it's only fair for non-KDE users to have some desktop eye candy as well.

—
Daryl Sorrell

Also try MGM, the “Moaning Goat Meter” if you can't get enough stats. —Ed.

Browsing SMB Shares

I read the *Linux Journal* article “It's a Cross Platform, All Right!”, by Marcel Gagné with enthusiasm and excitement that we finally have a very useful Linux SMB browser. Each and every time I would try to open any MS Windows folder in the SMB4K “Network” window the following message would appear:

```
An error occurred:

smbmnt must be installed suid for direct users mounts (1000,1000)
smbmnt failed: 1
```

—
Pete Mackies

Marcel replies: first of all, let me say “Thanks” for reading the column and for writing. I do appreciate it. As with a number of these projects, it is still in pre-1.0 development phase, but I had great success using the tool, both in building and in terms of use. Had there been huge problems, I would not have featured it. I've been so impressed with it that I've been using it as my Windows share browser since I wrote about it in the article roughly two months ago.

That's a problem with the smbmnt program and the permissions it allows, not specifically with SMB4K. There's a note about this in the FAQ on the SMB4K Web site (smb4k.berlios.de/faq.html). Specifically, the smbmnt binary is not SUID root and therefore doesn't allow anyone but root to mount. On my test system (Mandrake 10), the smbmnt binary is setuid by default.

You should be able to solve the problem you are experiencing by changing smbmnt's permissions. Start by doing an `ls -l` on smbmnt to make sure you aren't pointing to a different binary (as on my Mandrake system). Then, change the permissions as follows:

```
chmod +s /usr/bin/smbmnt
```

That should allow you to mount shares in non-root accounts.

I understand your concern, but I feel that when presented with a highly promising piece of software like SMB4K, it is sometimes worthwhile to mention it, despite its pre-1.0 status. Furthermore, I never write about something without having personally confirmed that it works. Unfortunately, I don't have access to every conceivable distribution, and it is possible that things don't always work perfectly on other systems. In the meantime, I hope you'll have some luck with the solution I offer. Thanks again for writing, and do take care out there.

Another Linux Cat

Here is a picture of my cat, Tosca, who seems to enjoy *Linux Journal* as much as I do. Often, when I arrive home with a fresh new *LJ* and sit down to read the many interesting articles, he'll come along and lie down on top of it on my table, and then fall asleep! He's also a big fan of Linux and specially the SolarWinds GL screensaver.

—
Kim Halavakoski

Bring *LJ* to Europe

I think it would be great to have a European *Linux Journal*. I think that there's a lot going on here that would give lots of interesting articles. Europe, besides having a larger population than the USA, has much more Linux aficionados, so I think the “gold mine” is here, and a Eurpean version of *Linux Journal* could be great.

—
Joï¿½ Vieira

Video Editing Software, Please

I've been a subscriber to *LJ* for more than a few issues now. Thanks for the great work and dedication. I get a kick out of at least one section per issue, so keep at it! I don't remember seeing any pieces related to Non-Linear Video Editing....One of my home hobbies is DVD production, and I'd really like to see some articles about the tools that are available in Linux for this.

—
Vasco Nï¿½oa

Missing Readers' Choice Category?

WAN adapters are no longer part of the *Linux Journal* Readers' Choice Awards. Why did you remove them?

—
Doug Vilim

We didn't get enough votes in this category. Check page 50 for a great Linux WAN story, and read *LJ* next month for the Readers' Choice results. —Ed.

Jump Pages for Articles—Good or Bad?

Would it be all right if I'm candid? I thought so. I have a couple of thoughts concerning your choice to move the on-line resources to your Web site. I got a subscription to a magazine, not to a Web site. Consequently, I want this to be self-contained. I don't want to have to go to the Web site to get the URLs.

The very first URL I went to (/article/7609) was not there. If you don't provide the resources as promised, you are doing a great disservice to your readers. It would be much, much better to use a little space on paper rather than to do this.

This could be okay, if you had user feedback like the wonderful feedback on php.net, which allows users to share their experiences. Then, this new approach to using the Web site would cause me to find instantaneous feedback like corrections and other tips that pertain to each article, and further, I would likely go to this page, as I would want additional information on the topic of the article.

It could be okay to not have to type in long URLs, but really that's only rarely an issue, and if you were truly concerned, I'm sure you could provide shortened

URLs. This is really just a lame excuse so you can get more traffic on your site.

—
Shane

We'll be candid right back then. We'd rather fit in one extra original article per issue than a chunk of URLs for all the other articles in the magazine. And, you need a browser and a Net connection to use the URLs anyway. The late jump page was bad, and our fault. We're going to adjust our Web and print calendars to fix that for future issues. In the future, the jump pages will be up before you see the issue. Try it now. The jump pages are set up for comments in a threaded Slashdot-like style just like the Web-only articles on linuxjournal.com. —Ed.

Bad Magazine CD, Bad, Bad

I have been rereading several magazines that I have bought over the last couple of months concerning Linux. At present, I am using Red Hat 8 as it is the only one I can get to operate on my laptop.

I recently purchased a copy of *Linux Format* from England with a copy of Mandrake 10 on the cover. It loaded on my laptop but I was unable to use it as far as the Internet goes—no drivers and no information concerning an external modem.

I am sure that many people who are fed up with Windows would very much like to try Linux, but with the extreme difficultly of finding drivers and not knowing if any particular external modem will work, we are defeating our purpose in trying to promote Linux to many people.

—
Herb Taylor

There are often better sources of Linux CDs than magazines. Here's some advice on picking a distribution: www.linuxjournal.com/article/4619. —Ed.

Layout Software for Books and More

It's been a long time since I've seen an article in *LJ* about LaTeX. With the now-mature LyX GUI front end, it's more than ready for prime time in business. LyX will not replace OpenOffice.org, for example, for simple letters, invoices and the like (although it can). But, for reports, articles, books, slide presentations and other printed materials, including PDF distribution, it cannot be exceeded. I'm using it for my book (to be published this autumn by Springer-Verlag) using

their svmono class file, so I can send it to them essentially camera-ready. LyX is much like Linux itself; use the half-hour tutorial and you're immediately productive, but the depth and breadth of what it can do is a never-ending learning process. The on-line support is outstanding, and the output, being typeset, is visibly better than anything done with a word processor, even on the same 600dpi laser printer.

—
Rich

Why Mozilla?

I read Mr McFarlane's article on cross-platform development in Mozilla (“A GUI for PS(1) Built with Mozilla”, July 2004), and much of his book (*Rapid Application Development with Mozilla*) as well. I salute him for his work explaining the Mozilla environment.

For all that I am a fan of Mozilla, but I must confess that I do not see its appeal as a development platform. The more I learn about development with Mozilla, the less I understand what it offers. As far as I can tell, Mozilla development requires that one master at least XUL (the Mozilla GUI layout language), CSS, JavaScript, RDF and the Mozilla template language (another XML dialect). In order to get your software working, you must also contend with a fairly arcane process of software installation, in which you must place your code, and RDF to describe it, in very specific places in the directory structure. Even after overcoming these hurdles, one is left with a fairly restricted set of interface tools. For example, if you want something that features free-form drawing, like a Canvas, you are out of luck. Finally, there are only the weakest of debugging tools to help you grapple with all of this machinery.

In contrast, consider Tcl/Tk, PerlTk or the Python equivalents (wxWidgets is a reasonable alternative to Tk as a widget set). In this case, there is only one notation to deal with and the debugging tools are good. In addition, these languages all feature some sort of read-eval-print loop for rapid prototyping and testing. Finally, all of these approaches are platform-independent. So what application development niche does Mozilla fill?

Please note that I am not asking this because I want to start a programming language war, nor because I want to denigrate the work of McFarlane and others! No, this is a serious question, and it's one to which I'd like to have a good answer. I am a fan of the Mozilla Project, and I have invested a fair amount of time in understanding the development process. I'd like to see a way to use this stuff, but right now the hurdles seem too high.

—
Robert Goldman

Nigel replies: I note your puzzlement and take no umbrage at such well-put remarks. Here's my brief take.

Mozilla negatives:

- We're only at version 1. There's lots of nasty problems in any version 1, and lots of missing items.
- XML learning curve for traditional programmers—it may not seem obvious, but there is a collection of people who just love stitching applications together using XML. The XML learning curve is also a toolset—one different from the traditional 3GL toolset. One can get a lot done with it.

Mozilla positives:

- Mozilla applications can run from chrome, local disk or across the Internet without change. That's powerful flexibility. The execution environment is widely deployed and highly secure. At least on the graphics side, Mozilla is better integrated in some areas with native GUIs than, say, Tcl/Tk.
- Linux lacks a Visual Basic-like easy-to-use type of environment. Mozilla aspires to a degree to achieve that.
- Mozilla development proceses are a natural extension of Web development processes; there's already an audience and a work flow.
- Mozilla uses no cryptic syntax. JavaScript, XML and CSS are all well-established standards. Niche syntax, like Python and Tcl, are unlikely to get as widespread support. That's not to dismiss those languages; they're good. It's just that they're also odd.
- Web delivery—with broadband increasingly available, Web delivery of applications is a real option for service-oriented businesses.
- Records management—there are few free 4GL-style tools available on Linux. Records management applications are a very large segment of the computer industry. Mozilla presents front-end opportunities for vertical apps. Using 3GL code to develop advanced 4GL apps is just not sustainable in the long term (I think).

This is my two cents. I agree that what Mozilla does can be done in other ways. I think that there are ways of leveraging Mozilla that are uniquely efficient, though.

Thrilling USB Story

I read Greg Kroah-Hartman's article on USB snooping [*LJ*, August 2004] with great interest, not because I'm interested in USB devices or CrypToken devices, but because of Greg's writing style. Learning how he thought, and the false steps he took, kept me reading the article. Greg clearly understands that articles not only need to live up to technical standards, but must be fun to read!

—
Jon Miner

Ultimate Linux Box Hardware

In the sidebar to the Ultimate Linux Box article in the August 2004 issue, you note that if there is any one thing to buy this year, it's an HDTV card. You also talk about doing articles next year on pre-ban cards. So, do you have any card suggestions? I may “kick myself” for not getting a card now, but I'll kick myself more if I spend a couple hundred dollars on a card that won't function under Linux. Thanks for any input, and thanks for a consistently great read.

—
Gar

Ultimate Linux Box hardware is listed in the on-line Resources page for the article at /article/7614. —Ed.

Photo of the Month; Penguin Hatching Season

At 9:25 pm on the evening of July 14th, Sebastian Phillip Gagné came into this world. He weighed in at 7 pounds and 14 ounces (3.565kg) and was just over 21 and a half inches long (55cm). This beautiful blue-eyed wonder also has a crop of thick luxurious brown hair. Both baby and mother are doing great! Attached is a picture that requires a little explanation...as you might expect, it was inevitable that somebody would give him something penguin-themed—in this case, a bib. This picture was taken on July 22, 2004.

—
Marcel Gagné

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

Archive Index Issue Table of Contents

Advanced search

UpFront

diff -u: What's New in Kernel Development

Zack Brown

Issue #126, October 2004

Wichert Akkerman has been trying for some time to add a **Debian package** build target to the kernel sources to complement the RPM target. His initial effort in October 2003 suffered the fate of bad timing, falling smack in the middle of the 2.6-test series, which had started in July and led to a 2.6.0 announcement in December 2003. Wichert got some work done on the patch at that time and apparently has been maintaining it privately until June 2004, when he tried again to submit it. This time, with the kernel still in some amount of flux at a recent 2.6.6 release (at the time of this writing 2.6.7 also has come out), Wichert's patch got a much better reception, and it looks as though a .deb target will be accepted into the kernel without much fuss.

For a long time, **Mel Gorman** has maintained on-line documentation of the 2.4 and 2.6 **Virtual Memory** Subsystem. After Linus Torvalds' surprise move to replace the subsystem en masse in the early 2.4 releases, documentation was scarce or nonexistent. Mel worked like a dog to describe its inner workings and to offer them on-line as a free resource, and his work eventually was picked up by Prentice-Hall for inclusion in the Bruce Perens Open Source Series. The book, called *Understanding the Linux Virtual Memory Manager*, is a down-and-dirty examination of one of the most difficult aspects of the Linux kernel.

The relationship between the kernel and the **C compiler** is one of constant controversy. When a kernel release won't compile, often it is not entirely clear whether the problem is with the C code in the kernel or with the compiler's attempt to convert that C code into machine language opcodes. Lately, **Linus**

Torvalds and others have begun to speculate that future versions of Linux, perhaps in the 2.8 or 3.0 series, would require a GCC version beyond 3.3. Linus cited bad handling of aliasing in earlier GCC versions, but the compiler people are likely to have their own views on what is broken in their code and what isn't. In any case, it's not unusual for the kernel to require a tool upgrade, but every time it happens, the break in compatibility opens the floodgates for requiring other tool upgrades. The flip side is that it is generally preferable to require no tool upgrades at all, so the user has as easy a time as possible upgrading the kernel when they want to.

Greg Kroah-Hartman has taken official responsibility for being the **sysfs** maintainer. He was forced out into the open by **Christian Gmeiner**, who had been trying to identify the sysfs maintainer to answer a coding question. Before then, no one was listed in the MAINTAINERS file, but Greg has said he'll submit a patch at some point to update that with his own contact information. The sysfs project has advanced quite far in the 2.5 and 2.6 time frame, and is now the preferred method to interface with the running kernel. Of course, some folks still use /proc and other mechanisms, but they gently are corrected and have been gradually (and sometimes quickly) coming around to the new ideas. sysfs has established itself quite successfully over the past couple of years.

Write support for **Microsoft's NT filesystem** (NTFS) is not exactly a holy grail of computing but, nevertheless, it has been a persistent itch that some programmers have felt a technical need or personal compulsion to scratch. **Anton Altaparmakov** is one of these and recently has announced a step in the "write" direction for NTFS: the ability to overwrite files that already are resident on the filesystem. Along with this, the NTFS driver also performs some housekeeping chores that previously had to be done by hand to prevent corruption. Clearly, NTFS is not yet ready for hard-core use and still is best for users needing to copy data out of their old disks, but it seems a time will one day come when full-on NTFS support will be available in Linux.

The **PC9800** sub-architecture is slated to be dropped from the 2.6 tree. The hardware is obsolete, which is not necessarily grounds for removing a feature from the kernel. But, in addition to that, the PC9800 code is unmaintained, which certainly is grounds for removing a feature. Linus Torvalds has made it abundantly clear that it doesn't matter even if there are users clamoring for a feature; if code is broken and no one will fix it, out it goes. Of course, thanks to the proprietary **BitKeeper** revision control system, it is easier now to put the code back into the kernel once a maintainer is found. And, among other justifications for putting unmaintained code on the chopping block is that it does get the attention of people who might then choose to maintain it.

Another bit of code that may end up being dropped from the 2.6 tree is the venerable **UMSDOS** feature. Back in the old days, UMSDOS was used to install Linux within an existing MS-DOS partition, so folks could experiment with it without repartitioning their drives. Nowadays, UMSDOS is not used as much, but its historical value is significant, so much so that Linus Torvalds, **Andrew Morton** and the other kernel maintainers might be happy to leave it in the kernel for posterity's sake. Unfortunately, however, UMSDOS is currently broken in 2.6, which greatly reduces the chances that it will be kept around. Without a maintainer to nurse it back to health, the tool that introduced many folks to Linux in the early days may become only a memory.

LJ Index—October 2004

- 1. Days in development for Linux 2.6.0: 680
- 2. Different changes accepted into the kernel source tree: 27,149
- 3. Average changes per hour during development period: 1.66
- 4. Number of different developers who contributed at least one kernel patch: 916
- 5. Number of different developers who contributed one kernel change: 413
- 6. Number of different developers who contributed two kernel changes: 117
- 7. Number of different developers who contributed three kernel changes: 57
- 8. Number of different developers who contributed four kernel changes: 38
- 9. Number of different developers who contributed five kernel changes: 20
- 10. Number of kernel changes contributed by the top ten developers: 10,933
- 11. Number of kernel changes contributed by the top five developers: 6,956
- 12. Average number of changes per day for each of the top five developers: 10
- 13. Number of merges in the kernel source tree: 6,175
- 14. Average number of merges per day in the kernel source tree: 9
- 15. Number of modifications per hour exceeded over the development period: 2
- 16. Millions of people in Sweden: 8.98
- 17. Millions of fixed phone lines in Sweden: 5.4

- 18. Percentage by which fixed phone lines in Sweden declined in 2003: 2
- 19. Milliions of cell-phone subscriptions in Sweden: 9.07
- 20. Swedish cell-phone penetration percentage: 100.1

- 1–15: Compiled by Greg Kroah-Hartman for *Open Sources*, shared on the Linux Elitists list
- 16–20: smh.com.au, sourcing *Dagens Nyheter* and the Swedish National Post and Telecom Agency

They Said It

It is wrong to say that demand creates supply. It is the other way around.

—Henry Ford, www.jyu.fi/library/elkok/isbn9513911675.pdf

Invention is the mother of necessity.

—Thorstein Veblen, www.quotationspage.com/subjects/invention

Well now home entertainment was my baby's wish So I hopped into town for a satellite dish I tied it to the top of my Japanese car I came home and I pointed it out into the stars A message came back from the great beyond There's fifty-seven channels and nothin' on.

—Bruce Springsteen

If we believe we “own” the customer we will quickly die.

—Juha Toivari, EVP, Head of Authentication and Certificate Services, Nordea Bank, IDman conference speech, London, July 2004

The technology that preserved the balance of our history—between uses of our culture that were free and uses of our culture that were only upon permission—has been undone. The consequence is that we are less and less a free culture, more and more a permission culture.

Lawrence Lessig, *Free Culture,* blogspace.com/freeculture/Introduction

Understanding the open-source process can generate new perspectives on very old and essential problems of social cooperation. And it can provide an early perspective on some of the institutional, political, and economic consequences for human societies of the telecommunications and Internet revolutions.

Steven Weber, *The Success of Open Source*, www.hup.harvard.edu/pdf/WEBSUC.pdf

Cynicism about new technologies shouldn't blind us to the way the business is changing. New ways of organizing emerge periodically—and can be of huge significance. Open source may well be among them.

—Matthew Lynn, Bloomberg.net, quote.bloomberg.com/apps/news?pid=10000039&refer=columnist_lynn&sid=aqq4dVXwLQ8Y

Archive Index Issue Table of Contents

Advanced search

On the Web

Creating Your Own Security

Heather Mead

Issue #126, October 2004

Don't rely on others to ensure your computer's security—it begins with you.

Although it may be tempting to blame faulty software, incomplete patches or inadequate monitoring when security is breached on the Internet, a network or a personal computer, we must remember the part we ourselves play in computer security.

Internet security is the focus of this month's issue; back in the April 2004 issue, we looked at application and intranet security. More articles discussing these and other aspects of security can be found on the *Linux Journal* Web site. For an overview of how Linux and open-source software are meeting the security requirements of government agencies, read “Open Source Innovation within the DoD” (www.linuxjournal.com/article/7644) and “GNU/Linux Clears Procurement Hurdles” (www.linuxjournal.com/article/7678). Both articles are part of Tom Adelstein's Web column, Linux in Government. Keeping track of government's standing on Linux and open source is important, especially considering the warning issued this past summer by one government agency—the United States Computer Emergency Readiness Team—“advising people to use a different Web browser”.

If you are looking to enhance security on your home or small office network, putting a firewall in place is a good start. But, you knew that and you probably already have one. How about something a little more interesting—something that could turn into a nice little DIY project? In “Building a Diskless 2.6 Firewall” (www.linuxjournal.com/article/7383), author Christian Herzog explains how you can salvage some minimal hardware, replace the hard drive with a CompactFlash (CF) card and employ BusyBox to build a machine with an “iptables firewall, SSH dæmon, DHCP server and DNS server”. Christian's tutorial walks you through choosing the right software, selecting the best

filesystem for the CF card, compiling the 2.6 kernel, filling the filesystem and booting with GRUB.

We all know that one important component of computer security is being prepared and able to recover when something goes wrong, as it always does on some level. A large part of one's ability to recover depends on the quality of the data backups, yet backups don't always rank high on people's to-do list. In fact, Phil Moses, author of “Open-Source Backups Using Amanda” (www.linuxjournal.com/article/7422), notes, “Data probably is the most important element in computing, but in too many cases I see data backups overlooked or approached in such a carefree manner that I shiver.” To this end, Phil focuses his article on Amanda, explaining how its ability to take on multiple configurations and multiple backup tape devices, “while requiring a minimum amount of time and resources” makes it an ideal backup solution.

Finally, for something a little different, check out Marco Tabini's article “PHP as a General-Purpose Language” (www.linuxjournal.com/article/6627). Using a PHP-based news aggregator script as an example, Marco demonstrates how the command-line version of PHP can “perform complex shell operations, such as manipulating data files, reading and parsing remote XML documents and scheduling important tasks through cron.”

Have you found a better way to monitor the traffic coming in and going out of your network? Discover a new use for your old 386? 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.

Peace from Mail Worms?

My question is: why would I want to use Linux? Right now, I'm using Microsoft Windows XP. Basically, I surf the Internet, send a little e-mail, download some music and occasionally work with Microsoft Word. I'm not a computer genius. Is Linux for me? Or, should I stick with what I'm doing and not worry about it? I recently became interested in learning about something different because of all the viruses, worms and spyware. Thanks for the help.

—
Henry Stone

henrystone39@hotmail.com

You can deal with many of your immediate security problems by visiting mozilla.org for a free Web browser, Firefox, and e-mail program, Thunderbird, that can run on your existing OS. It's best to switch to Linux when you are working on a project that motivates you to learn Linux, and it sounds like you're not there yet.

If you are considering moving to Linux in the future, be careful about downloading music on-line. Avoid copy-restricted formats. The two top vendors of copy-restriction systems for music also are in the OS business and don't support Linux. Stick with formats such as MP3, Ogg Vorbis and FLAC, which are not copy restricted, and you will be able to move your music to your new computer when you upgrade.

—
Don Marti

info@linuxjournal.com

This is an excellent question, and one that should be considered by anybody before installing any operating system. An operating system is only as good as the software you run on it. If your current platform serves you well, there's no reason to switch. Consider the following issues:

- Do you need portability? Linux can run on a number of different architectures, so you theoretically could buy an Apple laptop and run the same environment.
- How secure do you need to be? Linux has a low patch release rate, and most patches do not require a reboot. Linux systems also rarely are the targets of automated attacks. This does not mean they are secure, but it does mean they can be cheaper and less time consuming to keep secure.
- What are your skill sets? If you aren't comfortable with Linux, do you care enough to get comfortable? Polished distributions and functional desktop interfaces have made Linux much easier to learn, but if you have neither the time nor the interest, it's not worth the effort.
- Do you require licensing flexibility? Linux can be purchased from a commercial vendor or downloaded for free. If you decide you don't like your current vendor, you can choose another source and be assured of being able to run the same applications you did before.

Almost any operating system can meet the basic task requirements you listed. If your current environment meets those needs, and you don't have security or flexibility concerns, there's no reason to switch. But, if you would like to explore a more secure or flexible operating system, Linux is an excellent option to explore.

I advise all new Linux users to take a look at the Knoppix distribution (available at www.knoppix.net). This option does not require installation; it simply runs off the CD. Because of this, it does not run as fast as a distribution installed on your hard drive, but it makes no changes to your system. You thus can test Linux, OpenOffice.org, Mozilla and other applications and decide whether you like them before making the switch.

—
Chad Robinson

chad@lucubration.com

Standard for Blade Servers?

Is there a standard format for blade server hardware? Right now, I have a variety of tower and rackmount servers using primarily the ATX format. I'd like to consider a blade server to save space, but I am concerned about being

locked in to one vendor, especially with the rapid turnover of hardware vendors these days. Do blades from one vendor fit in another vendor's blade server frame? I don't seem to receive good responses from vendors on this question.

—
Michael Barnes

mbarnes@hcjb.org

Your question highlights the state of the industry with respect to blade servers and one of the chief customer complaints. Blade servers are relatively new, and as of yet, no standards exist for backplanes or blade formats. Every vendor has its own proprietary format. This may change over the next few years, but don't hold your breath. Blade server vendors typically consider the backplane/chassis a nominal cost component in a blade server infrastructure. That means little incentive exists for them to band together to produce a common format. Each backplane is typically from 3U to 8U tall and can fit anywhere from 8 to 20 blades.

Multiple backplane chassis from different vendors can coexist in a single rack, so if you wanted to change vendors, typically you can start installing new chassis. In fact, blade server vendors may not support specific backplanes for a long time, so even if you select a single vendor, you may have multiple chassis types in a single rack.

—
Chad Robinson

chad@lucubration.com

Fedora Install Chokes

I'm writing because I'm having a problem installing Fedora 2. I don't get very far before the system reboots. On May 28, 2004, I downloaded and burned four CD-ROM ISO images from one of Red Hat's mirror servers, download.fedora.redhat.com/pub/fedora/linux/core/2/i386/iso. The ISOs are timestamped May 13th, as I recall.

The disks seem fine. When I boot from CD1, I see the standard welcome screen with the Fedora 2 Core logo, <F1> through <F5> options and directions to `press enter to begin a graphical installation, or type`

`linux text to begin a text installation`. It's at this point that I encounter problems. I've tried the following, all without any success:

- No options; simply press Enter for graphical install
- `linux text`
- `linux noprobe`
- `linux askmethod`
- `linux skipddc`
- `linux nofb`
- `linux resolution 1024x768`
- `linux mediacheck`

All of these options fail and result in a system reset when the following occurs:

```
loading vmlinuz.

loading initrd.img.
```

The only success I've had is when I use the memtest86 option. It kicks in with a memory test program that has run for hours without incident.

I also have VMware, so I thought I'd give Fedora 2 a try as a VMware virtual machine. It works fine, but it doesn't suit my purposes. I'd like to install Fedora 2 on a dedicated hard disk. The noticeable differences between my system and a VMware virtual machine are the following: the VMware BIOS sees only one IDE disk, a virtual disk that I believe Linux refers to as /dev/sda. My actual hardware is as follows:

- BIOS: American Megatrends v2.51
- Motherboard: Asus P4P800 Deluxe motherboard, Intel P865PE chipset, Socket 478
- CPU: Intel Pentium 4 2.4GHz
- Memory: 256MB PC3200 DDR RAM (only one module, so it runs at 400MHz)
- Hard Disk: Primary Master—EIDE 20GB hard disk, Secondary Master—Samsung DVD/CDRW combo drive and Third Master—SATA 80GB hard disk
- CD-ROM drive: Samsung DVD/CD-RW combo drive
- Graphics Card: ATI RADEON 9200se 128MB AGP 8x

Despite my BIOS supporting over-clocking, this feature is turned off. Thinking it was a BIOS tweak that caused the problem, I chose `Load Setup Defaults` but saw no change.

When I attempt to install Fedora 2 in the described manner, I'm attempting to install to the 20GB IDE hard disk on the primary master IDE interface. As this configuration failed, I've also attempted to install to the 80GB SATA drive on the third master IDE interface. Each time, I physically disconnected the other hard drive from the computer and removed the entry in BIOS so there could be no confusion.

All attempts have failed. I am led to believe that this is a BIOS problem or a motherboard incompatibility. I am unable to investigate this matter further, as I have reached the limits of my computing knowledge. I'm reluctant to try a BIOS upgrade unless I know for certain that this is the problem. I've upgraded BIOS chips before, each time with my heart in my mouth. My BIOS upgrade path now is failsafe, according to the motherboard manual, but I'm still reluctant.

The 20GB IDE disk comes from an old Cobalt Qube, and I've had netBSD installed on it for some time, running as a backup server in the Qube. I've installed other versions of Linux, including Debian Woody and LindowsOS 4.5, on this 20GB drive when connected to the computer configuration I've described above, all without incident.

Finally, I've not tried the pen-boot option described in CD1's README-en.html file, as I don't have a USB pen drive. I built my machine from components and don't have a floppy drive, so booting from floppies also is out of the question.

I've queried the Fedora support forums, which initially asked me to do a `linux mediacheck`. As the system resets immediately after the `loading initrd.img.` message, it wasn't possible to test the disk set on my system. But then I had the idea of doing the `linux mediacheck` on a VMware virtual machine, and the disks check out fine.

I'm beginning to think that this is some kind of bug, either in my BIOS or in Fedora 2 Core's initrd.img file. I don't know what to do next; I don't want to bother the developers, who generally refer one to the support forums anyway. I'm at a loss, please help!

—
Johnny

johnny@skydiveflorida.net

You've done a great job of doing the detective work here. I bet there is some incompatibility between the FC2 kernel and your hardware, as you have alluded to. I had the same problem with booting FC2 on an Epia 800. You may want to

try to find a FC2 boot disk to boot your hardware in a more compatible mode. I was able to find one doing a search at the Bugzilla interface for Fedora at fedora.redhat.com.

—
Christopher Wingert

cwingert@qualcomm.com

Try `linux noacpi`, `linux disableapic` and `linux noacpi disableapic`. Some motherboard and BIOS combinations cause problems with the APIC and ACPI functions in Linux, and disabling them always is a good option to try.

—
Chad Robinson

chad@lucubration.com

Balancing between Two Interfaces

I have a Linux proxy server. I have two connection methods to the Internet, wireless and a DSL modem. How can I configure my proxy server to use both methods to visit the Internet, so that if one of them is down, I still can get to the Internet without changing settings? Now I am using two proxy servers; if one link is down I turn on the other one.

—
John Van De Veer

jvdvjav@hotmail.com

Simply use the following command:

```
/sbin/ip route add default equalize nexthop \
via A.B.C.D dev eth0 weight 1 \
nexthop via E.F.G.H dev eth1 weight 1
```

Replace the *A.B.C.D* with the peer IP address of your connections and the eth0 and eth1 with the actual devices used. Doing so balances the traffic by informing Linux it can use both routes with the same weight.

—

Mario Bittencourt

mneto@argo.com.br

New Products

UniComp keyboard, NetDirector Linux Configuration Suite, Enea Orchestra and more.

Linux 101

UniComp, a keyboard product and service company that offers customized keyboards based on classic IBM designs, announced the availability of the Linux 101 keyboard. The Linux 101 is a programmed keyboard that rearranges the Ctrl, Caps-Lock and Esc keys for a more convenient layout for Linux users. The Linux 101 keyboard is available with a buckling spring mechanism, like the IBM Model M keyboard from which it descends, or with a quieter rubber dome design. PDFs of each design layout are available on the Unicomp Web site.

UniComp, Inc., 510 Henry Clay Blvd., Lexington, KY, 40502, 800-777-4886, www.pckeyboard.com.

NetDirector Linux Configuration Suite

The NetDirector Linux Configuration Suite, from Emu Software, is a scalable, multiserver management tool designed to work in a heterogeneous Linux environment comprising a network of servers from different manufacturers running multiple computing services. The NetDirector Suite's GUI allows administrators to manage centrally the configuration of servers running applications such as Apache, Samba, DNS, DHCP, e-mail and FTP. Servers can be grouped and managed together according to organization, geographic region or server application. Policies then can be applied to entire groups or specific servers within the organization.

Emu Software, Research Triangle Park, North Carolina, 919-313-5186, wwww.emusoftware.com.

Enea Orchestra

Embedded Linux and Enea's hard real-time operating system, OSE, are the basis of Enea Orchestra, an integrated software platform for high-availability telecom and datacom systems. Enea Orchestra enables telecom and datacom OEMs to deploy distributed fault-tolerant, high-availability software solutions across multiple processors and blades. Orchestra also includes embedded development technology from Metrowerks Corporation to simplify kernel debugging, board bring-up and application creation and testing. Resulting applications can run under any Linux distribution, OSE or any combination of the two. Enea Orchestra is available on a subscription basis and includes the application development suite and the platform development suite.

Enea Embedded Technology, 12760 High Bluff Drive, San Diego, California 92130, 858-720-9958, www.enea.com.

Motorola A780

The Motorola A780 mobile phone is based on Linux and Java software. Using the flip-phone form factor, the A780 features a PDA-like quarter-VGA color touchscreen, Bluetooth networking and synchronization, a 1.3 megapixel digital camera, MP3 playback, 48MB or removable TransFlash storage and more. The A780 is a quad-band GSM phone, so it can support common US and international bands. The A780 also features EDGE, enhanced data rate for global evolution, a data networking technology capable of supporting Internet access speeds of up to 240Kbps.

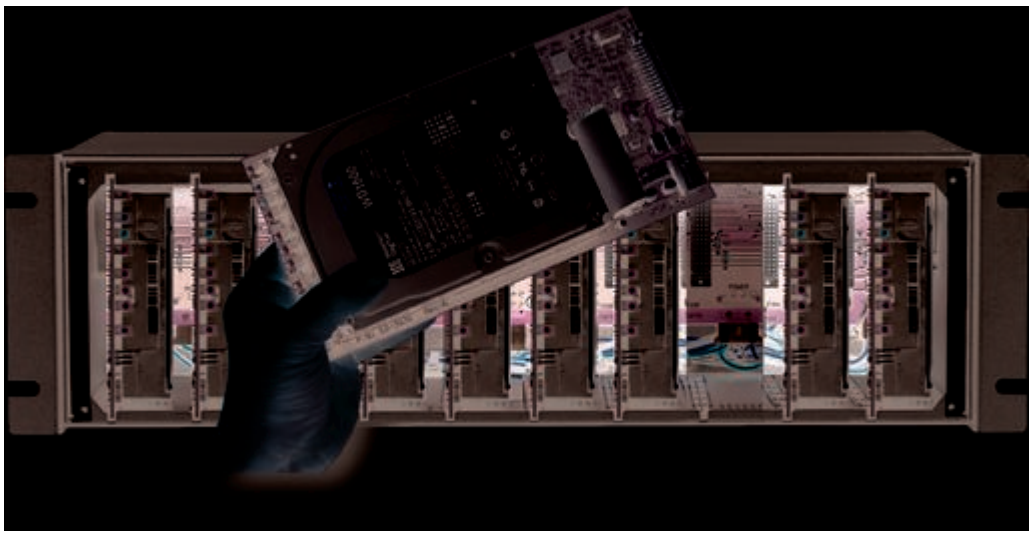
Motorola, Inc., www.motorola.com.

EtherDrive Storage Blades

Coraid, Inc., announced the release of EtherDrive Storage Blades, a scalable networked block storage solution for servers that uses the ATA-over-Ethernet (AoE) protocol. EtherDrive Storage Blades integrate standard ATA disk drives into a flexible rackmounted storage appliance. Each EtherDrive includes its own Ethernet connect and nanoserver to provide protocol conversion from Ethernet

to ATA. EtherDrives provide shared storage pools from 250GB to more than 16 petabytes. They are available with standard 2.5" or 3.5" ATA disk drives. Source code GPL drivers for Linux 2.4 and 2.6 kernels are available, with other OS drivers becoming available soon.

Coraid, Inc., 565 Research Drive, Athens, Georgia 30605, 877-548-7200, sales@coraid.com, www.coraid.com.



Quadputer-Navion

Microway's Quadputer-Navion is an SMP server built with four AMD Opteron 850 model processors that runs SuSE Linux. Housed in a 4U chassis, the Quadputer-Navion includes an 810 Watt-redundant, hot-swap power supply and up to 18 SATA/IDE or five SCSI hard drives. The processors operate at 2.2GHz and feature HyperTransport technology. The 2.5" devices can be used to create an internal RAID system with a total storage capacity of up to 1.36TB. The configuration also includes Microway's Nodewatch and MCMS hardware/software management tools for remote cluster monitoring and control.

Microway, Incorporated, Plymouth Industrial Park, 12 Richards Road, Plymouth, Massachusetts 02360, 508-746-7341, www.microway.com