Linux Journal Issue #111/July 2003

Features

Running Linux on the Xbox *by Michael Steil*
The Xbox is basically a PC, so with a little work you can upgrade it to run your OS of choice.

AMD64 Opteron: First Look *by Michael Baxter*
Discover the new architecture that's backward-compatible with the x86 and has IBM, Cadence and others already offering products.

Network Management with Nagios *by Richard C. Harlan*
The servers are from many vendors, the management software budget is small and the demands are high. Find out how the team at John Deere made it work.

Indepth

Getting to Know Mono *by Julio David Quintana*
Working code shows how you can already work with objects created in one language, from another.

How to Index Anything *by Josh Rabinowitz*
Create a local search engine to search HTML and every other document format on your system.

wxWindows for Cross-Platform Coding *by Taran Rampersad*
A fast, stable toolkit for apps that run on any OS with a native look.

Running Linux on the Xbox

Michael Steil

Issue #111, July 2003

Modifying an Xbox can increase your PC hardware knowledge and provide you with a useful little system.

In November 2001, Microsoft entered the video console business with the Xbox, a machine that continues to outperform all other consoles in terms of processor speed and video performance. As with the SEGA Dreamcast, hackers started to port Linux to the Xbox in May 2002. Only three months later, the first kernel messages from an Xbox running Linux were published on the Internet. Now, a year after the start of the project, Linux runs reliably on all versions of the Xbox, and Xbox Linux is ready for daily use.

Xbox Hardware

The Xbox is driven by a 733MHz Intel Celeron processor and contains 64MB of DDR RAM (shared with video), an NVIDIA GeForce3 graphics processing unit (GPU), an 8GB or 10GB hard disk, a DVD-ROM drive, Ethernet connectivity, four USB-style controller connectors and TV-out (Figure 1 lists the details). This hardware overview sounds more like the description of a decent PC than a gaming console. The Xbox does not merely contain some typical PC components, such as an Intel CPU or an NVIDIA GPU, it actually *is* a PC in a smaller black case, with minor modifications. The Xbox chipset consists of the NV2A Northbridge and the MCPX Southbridge, both from NVIDIA. The NVIDIA nForce chipset for PCs is almost the same as the Xbox chipset. Its Southbridge IC is labeled MCP and contains exactly the same functionality as the MCPX: two USB controllers, an IDE controller, an Ethernet device and AC97-compatible Dolby Digital sound.

Figure 1. The Xbox hardware in a tree view. The numbers on the right are the PCI locations and IDs.

The background of the Xbox is simple. Because Microsoft already had an operating system, system libraries and the DirectX libraries for the PC, they decided to build the Xbox based on this well-known architecture. Initially, Microsoft wanted AMD to produce the CPU and the chipset for the Xbox; the video chip would come from NVIDIA. But Microsoft later changed its mind, switching to Intel for the CPU. So NVIDIA licensed the chipset from AMD, manufactured the ICs for the Xbox and sold the same design as nForce for the PC market.

The similarity of the Xbox to a PC not only made the process of installing and running Linux a lot easier, it made a lot more sense for people to use the Xbox as a computer. Unlike Dreamcast, PlayStation 2 or the GameCube, the Xbox always is equipped with a hard disk and Ethernet. And the PC hardware also makes it possible to use standard Linux distributions on the Xbox, with minor modifications.

Because of its price and its compactness, an Xbox running Linux can be used as a desktop computer (see Figure 2) or a server, replacing a standard PC, and because of its TV connectivity, it also can be used as an entertainment device for watching video or listening to audio.

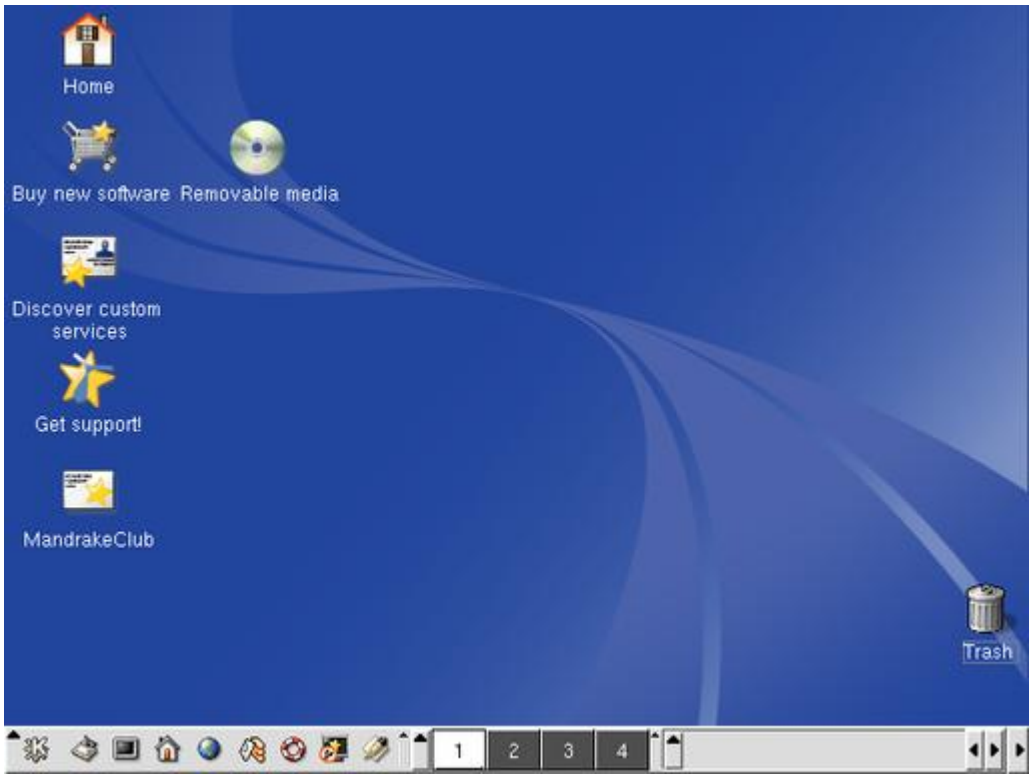


Figure 2. A KDE Desktop at 640 × 480 on an Xbox Running Mandrake Linux 9.0

Security

Despite the similarity of the Xbox to a standard PC, installing Linux on an Xbox is not simply a matter of inserting an installation CD. For one thing, the Xbox

boot process is a lot different from a PC's. PCs have a PCBIOS (basic I/O system) in ROM, which contains 16-bit library routines for keyboard, video and hard disk I/O, as well as a simple bootloader that reads the first sector from a storage device and runs it. The Xbox has no such BIOS. Its 256KB ROM image contains a statically linked, stripped-down, Windows 2000-based kernel, which runs the moment the Xbox is turned on. The hard disk—which is locked by an individual ATA password, so it cannot be read when connected to a computer or replaced with another hard disk—does not contain any operating system components. When the Xbox kernel is started, it unlocks the hard disk and tries to run the default.xbe file from a CD or DVD. If such a file cannot be found, it runs xboxdash.xbe from hard disk. This is the system configuration and audio CD player application permanently stored on the hard disk.

These .xbe files are executables, which are a lot like Linux ELF files, except they are signed digitally with Microsoft's 2048-bit RSA key. Changing a single byte within the file makes the signature invalid, and the file will be rejected by the Xbox kernel. Because of the lack of Microsoft's private key, the Xbox Linux Project cannot reproduce a valid signature; thus, we cannot create executables accepted by a standard Xbox. Two approaches are possible to get your own code running: replace the ROM or find a game with a bug that can be exploited.

The standard way for most people to get Linux running on an Xbox is to open the box and install a replacement ROM chip that overrides the onboard ROM chip. This so-called modchip can contain either a hacked version of Microsoft's ROM, which has the signature test, the hard disk test and some other things disabled, or a clean-room ROM implementation that gives the Xbox the personality of a regular PC. Although Xbox Linux supplies a bootloader that makes Linux run on hacked Microsoft ROMs (which Linux sites do not supply, but can be found on the Internet), the use of the Xbox Linux Project's clean-room implementation, called Cromwell, is recommended for legal reasons. The Cromwell ROM does not run Xbox games.

Figure 3. The Xbox dissected: the Philips DVD drive is on the left, and the Seagate hard disk is on the right. The green board in the background is a modchip that is, in this case, connected to a computer's parallel port.

Modchips that replace the onboard ROM are available from many video game hardware stores on the Internet for about $50 US. The first generation of modchips had to be soldered into the Xbox board parallel to the original Flash chip, which required about 30 wires. Second-generation modchips were connected to the LPC bus on the Xbox board, and they typically required only nine wires. Current modchips can be screwed onto the board without any soldering. They usually ship empty and can turn themselves off completely, so if you use the Xbox Linux Clean BIOS, you still can run Xbox games.

Because the original ROM contents are stored in a reprogrammable Flash chip on the Xbox board, it also is possible to overwrite the Flash contents in order to have a permanently modded machine, without installing any additional hardware devices. This can be done by installing a modchip, bridging two pairs of points on the board to disable the write protection of the Flash IC, running Linux, disabling the modchip and, finally, running an application called raincoat in Linux to reprogram the onboard Flash. Now, the modchip can be removed permanently, so you can use one modchip to convert a lot of Xboxes to Linux.

Recently, an anonymous researcher found an exploitable bug in the Electronic Arts game *007 Agent Under Fire*. In a post on an Xbox forum, he explained how to use a modified saved game to run the Linux bootloader. By connecting the write-protection bridges on the board, this method can be used to reprogram the onboard Flash within a Linux instance that has been started by this

modified saved game, without even temporarily installing a modchip. This is the cheapest and most simple way to make an Xbox Linux-compatible.

All these methods apply only to Xbox consoles that have been on the market to date. Microsoft keeps changing the Xbox design. By the time you read this article, a new board layout of the Xbox might have the LPC bus or the reprogrammability of the onboard Flash removed. Refer to the Xbox Linux web site for the latest information on this topic.

USB

Having a modded Xbox with either a modified Microsoft BIOS or the Xbox Linux BIOS means you can start the Linux installation—but how do you interact with the installer? The Xbox does have USB connectivity; all controllers, memory units and the Xbox Live headset are standard USB 1.1 devices, but with different connectors. By attaching an adapter, you can connect any USB keyboard, mouse, webcam, printer or scanner that is supported by the PC version of Linux. These adapters are sold on the Internet for about $10-$15 US, but it also is possible to build your own with little effort. All you need is an Xbox controller extension cable, which has the connector to plug in to the Xbox on one side, as well as a USB female connector. Both cables contain four wires in different colors; simply cut both cables and reconnect the wires to create a USB-female-to-Xbox-male adapter cable.

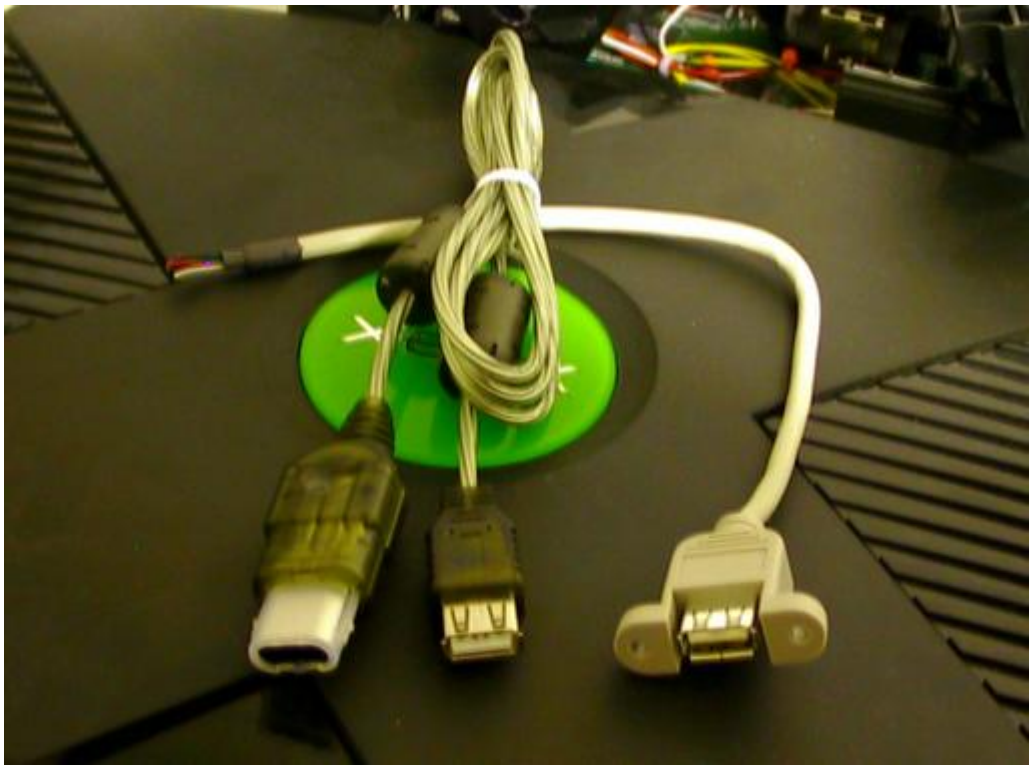


Figure 4. A USB adaptor cable can be built easily by using an XBox controller extension cable and the USB female connector on the right. You also can buy prebuilt cables like the one on the left.

The slot for the memory unit on your controller is another USB port, so you also can solder the USB female connector there.

If you plan to buy a USB keyboard and mouse for the Xbox, try to get a keyboard with a USB or PS/2 mouse connector built in. This way, you need only one Xbox-to-USB adapter. Macintosh keyboards have proven to work very well —and they don't have Windows keys, either.

Distributions

With a modded Xbox, a keyboard and a mouse, you now can choose either to build Xbox Linux yourself or to take one of our prebuilt installations. At the moment, Xbox Linux offers three main distributions: Mandrake, Debian and the Xbox Linux Live System. The latter is a version of Linux without X but with Trolltech's Qtopia. It does not install into the hard disk, and it can be controlled with an Xbox controller. It is supposed to give newbies an impression of the possibilities available with Linux. Mandrake and Debian are full distributions that install into the hard disk. Mandrake 9.0 is available now, and 9.1 will be available soon. Both are based on the PC versions of Mandrake Linux and are 100% compatible with them and their RPM packages. They contain GNOME, KDE and many popular applications. The Xbox version of Debian can be booted into X from CD, but it also can be installed to hard disk. Debian has smaller release cycles and is updated more often, therefore it is used by most developers.

Bootloader

All those who don't want to use the prebuilt distributions but want to do everything themselves, and those who want to know how it works, should have a look at the Xbox Linux bootloaders, the kernel and the XFree patches.

As pointed out earlier, Xbox Linux can be booted either through the Xbox Linux Clean BIOS or from an unsigned .xbe file that pretends to be a game and starts Linux on a modified Microsoft BIOS. Both these applications are based on the same bootloader code, which loads the kernel and initial RAM disk from CD/DVD or hard disk, reserves 4MB of RAM for video and passes 60MB to the Linux kernel. Details about the kernel and initrd filenames and locations are read from the file linuxboot.cfg, which must reside in the same directory as the .xbe file in the case of the Microsoft BIOS. In the case of the Clean BIOS, the user can choose from where to read it.

Kernel Patches

In order for a Linux kernel to work on the Xbox, only one byte has to be changed—but for a perfectly adapted kernel, you need more changes. The

Xbox has a severe bug in its PCI chipset that makes the machine crash when Linux tries to enumerate the PCI devices at boot time. By narrowing the region of PCI device numbers that are checked, this crash can be avoided. Another issue is the timer frequency: the Xbox system timer is off by about 6%, making music play too fast and making the kernel report the CPU clock frequency incorrectly.

The shutdown and reboot sequence is handled differently on the Xbox, but by adding some code to the kernel, it is able to reset and shut down the machine properly. Because copy restrictions often have been circumvented on other gaming consoles by inserting a legitimate game first and quickly replacing it with a copy, the Xbox, by default, reboots if you press the Eject button. Software can avoid this by replying to a request sent by the PIC16L chip that says not to reboot. Another patch in the kernel takes care of this. Yet some other code patches the report of the DVD drive, which pretends not to support video DVDs, although it does. On the Xbox Linux site, you can download patches for some well-known kernel versions that include all these modifications, plus support for the Xbox hard disk partitioning scheme and the Xbox hard disk filesystem (FATX). The modified files also are available in the Xbox Linux CVS. You have to enable the option for Xbox support in the kernel configuration.

Xbox Linux runs well in VESA framebuffer mode; that is, the bootloader sets up a fixed graphics mode and Linux inherits it, always writing into video memory but never accessing the actual video hardware. Alternatively, a patched version of the rivafb accelerated framebuffer driver is available, which allows changing the console video mode at runtime. In any case, you have to enable a framebuffer driver in the kernel configuration, because Xbox Linux does not support text mode yet.

With a minimal patch for the ALSA sound system, again available from the Xbox Linux CVS, the Xbox sound hardware can work with the i810 driver. The binary-only network driver for the NVIDIA nForce card, which you can download from the NVIDIA web site, works on the Xbox without any modification. An SMBus driver is needed if you want to enable the eject fix or to access the 256 bytes of EEPROM on the motherboard. You can use either the i2c-xbox module in the Xbox Linux CVS or the amd-756 module from the lm_sensors project. Both work equally well.

XFree86

XFree runs out of the box if you use the framebuffer driver and turn off PCI enumeration in the configuration file. A modified version of the nvdrv driver provides video mode change at runtime and 2-D acceleration (GLX extensions). Multimedia applications can render their window into an off-screen buffer, and

the video hardware stitches it into the visible screen when displaying it, scaling it in hardware. Precompiled versions of the driver are available. nvdrv is the open-source driver for NVIDIA graphics hardware, which does not support 3-D acceleration. Efforts are underway to patch the binary-only, 3-D-aware XFree driver available from NVIDIA.

Optimizing Applications

The Xbox hardware details are quite impressive, enough for playing DVD or DivX video in Linux. But for optimal performance, you should try to optimize the compilation of your applications for the actual hardware. The machine's Celeron is a Pentium III class CPU, and it supports the 686 instruction set, as well as MMX and SSE. Applications, including mplayer, detect this automatically. If you use the nvdrv XFree driver, you can enable GLX support for video applications. mplayer, for instance, is fastest in X with the nvdrv driver, even faster than it is in framebuffer mode. Also, keep in mind that you should decrease the hardware resolution instead of making the application scale the video output. In 640×480 mode, the PlayStation emulator epsx runs quite well with the picture scaled to 400 × 300 pixels.

Figure 5. A German data center is using an Xbox-based Domino Server, running a clustered environment with Domino on an IBM pSeries. The Xbox is the small system on the left.

Although the Xbox is equipped with only 64MB of RAM—which can be extended to 128MB with excellent soldering skills—desktop environments, such as GNOME and KDE, and applications like OpenOffice.org run quite well. With the help of VMware, you even can use MS-DOS and Windows 95/98/NT/2000 on the

Xbox. With a minimal X window, no desktop environment and no window manager, you can run Windows with up to 48MB of RAM.

Conclusion

With 1:1 ports of common Linux distributions for the PC and all major Linux applications running on the Xbox, it is ready for use on a desktop computer, a server or a multimedia device. With its excellent hardware and PC compatibility, there is more than simple hack value to it.

Resources

Michael Steil is studying computer science at the TU Muenchen, Germany. He initiated the Xbox Linux Project in May 2002 and is maintaining it. He can be reached through his web site, www.michael-steil.de, or via e-mail, mist@c64.org.

Archive Index Issue Table of Contents

Advanced search

AMD64 Opteron: First Look

Michael Baxter

Issue #111, July 2003

The Opteron architecture maintains binary comatibility with x86 but increases available address space to handle large applications, such as electronic design automation.

In advance of the release of the AMD64 Opteron processor, we had the opportunity to test a dual-processor SMP system with a 64-bit Linux distribution. As far as most Linux users are concerned, Opteron is the most significant new hardware introduction so far this decade. This early look covers the following key areas:

- An overview of the AMD64 Opteron architecture.
- A reference two-way SMP system from Newisys, Inc.
- Some results from running several GPL applications.
- A survey of early system performance estimates.

AMD64 Opteron Processors

The Opteron is a new device family based on a new 64-bit architecture that is compatible with the pre-existing x86 32-bit architecture. AMD's choice to preserve compatibility has positive implications for transmigrating 32-bit workloads.

The Opteron architecture supports four application programming models. The first is the General-Purpose model, which performs basic operations like memory access, control flow, exception handling and I/O. The General-Purpose model also sets up memory optimizations that are used by the other application programming models. The next model is 128-bit Media Programming, which uses 128-bit XMM registers. Operations supported under this model include integer and floating point on vectors and scalar data. Similar capabilities are supported in the 64-bit Media Programming model. The last

programming model is called x87 Floating-Point Programming, which uses x87 registers for 80-bit floating-point and scalar operations.

The code name for the processor core is Sledgehammer, and the device ships in a 940-pin ceramic micro PGA package. The current Opterons use nearly 106 million transistors in a 130nm Silicon on Insulator (SOI) process. The devices were fabricated at AMD's Fab30 in Dresden, Germany. The L1 cache has a 128KB capacity, split into a 64KB instruction cache and a 64KB data cache. An on-chip L2 cache has a 1MB capacity. The processor runs at 1.55V and provides a die size of 193mm2. The Resources section has more information on AMD64 architecture and open-source software support for this processor.

The Opteron processor is a highly integrated processor with features designed to attain balanced system performance. As such, it contains an integrated high-performance coupling link called HyperTransport, which offers 6.4GB/sec full-duplex data exchanges between processors or other HyperTransport nodes. Support is provided for up to three HyperTransport links, for a total of up to 19.2GB/sec peak bandwidth per processor. In addition, each Opteron contains an integrated memory controller, which offers very high bandwidth and error control capabilities. ECC (error correcting code) protection is provided for L1 cache data, L2 cache data and tags, and in external DRAM with hardware scrubbing of all ECC-protected arrays.

AMD has a three-digit part numbering scheme for the Opterons. The first digit indicates the intended SMP scalability, which is two-way in the 1.8GHz Opteron Model 244 and the 1.6GHz Opteron Model 242 covered in this article. The second digit indicates relative performance within the scalability family. As chip manufacturing costs decline and process technology improves, other model numbers of a given scalability class will emerge at higher frequencies and lower costs. AMD also will be manufacturing a one-way Opteron, which is intended for high-performance lower-cost systems.

Models 240, 242 and 244 are available at the time of this writing. The eight-way capable models 840, 842 and 844 are scheduled to be available in May 2003 and model 144 in Q3 of 2003.

The Opteron extends the x86 architecture, allowing customers to run existing 32-bit applications on a 64-bit OS. Customers who run a 64-bit OS will be ready to support future 64-bit applications and migrate at their own pace, while maintaining the usefulness of their 32-bit applications.

Newisys 2100 Server

The reference platform is called the 2100 and was realized by Newisys, Inc., a technology provider that now has two years of experience with Opteron (see Resources).

Figure 1. The Newisys 2100 puts two Opteron processors in a 1U enclosure.

The 2100, a 1U, two-processor, rackmountable system, is superbly engineered. The mechanical and electrical design offers reliability in a dense package (Figure 1). For example, the power supply actually is designed for 500,000 hour MTBF. If you work a typical 2,080 hours a year, this level of reliability would be like working for 240 years without making a mistake.

With Opteron-balanced chipsets and excellent board-level integration features, this system has improved memory performance and capacity significantly. The result is a high-performance balanced server design with robust I/O. The evaluation system came with 6GB of PC2700 memory, but the server supports 16GB.

Figure 2 is a top view of the system. The two copper heatsinks are the processors. Two speed grades are supported: the 1.6GHz Opteron model 242 or the 1.8GHz Opteron model 244. The Opterons link to each other and to the chipset using HyperTransport. The CPU-to-CPU bandwidth is 3.2GB/sec—in each direction (full-duplex). The Opterons each have an internal memory controller that supports ECC DDR SDRAM at a bandwidth of 5.33GB/sec (each). There are two banks of memory, one next to each processor.

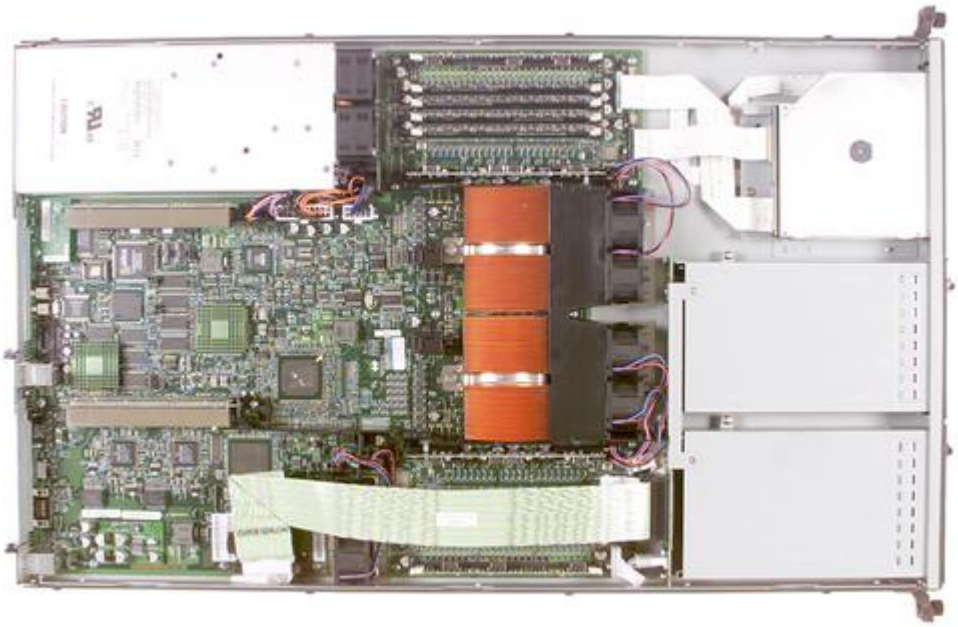


Figure 2. A Top View of the Newisys 2100

The AMD-8000 HyperTransport chipset includes an AMD-8131 HyperTransport PCI-X chip, as well as an AMD-8111 I/O hub chip. The AMD-8131 is configured to drive a full-slot PCI-X 64-bit/133MHz at 1GB/sec data rate, as well as a half-slot PCI-X 64-bit/66MHz, at a 0.51GB/sec data rate. The AMD-8131 also provides a pair of triple-mode NICs (10M/100M/1GB), plus a dual Ultra-SCSI RAID controller. Our test system had dual hot-swappable Ultra-SCSI drives in a RAID configuration in the front of the case. The AMD-8111 chip provides a VGA port, IDE CD-ROM and a USB port. Separately, a SuperI/O chip provides a floppy, keyboard, mouse and conventional serial port.

Figure 3. The server management processor has its own LCD display, under the CD-ROM drive.

The system also contains a separate embedded server management processor and it runs Linux. This subsystem is based on a Motorola XPC855T PowerPC processor, running kernel 2.4.18. In addition to a small front-panel control console, the server management processor provides a pair of isolated 10/100 Ethernet interfaces to connect to an independent management subnet. Thus, system management can be done without a keyboard and monitor, or even a serial console access server, and using it does not take up one of the PCI slots.

This system's management facility is quite advanced. The management processor supports SNMP, CIM and IPMI protocols. NIS, Microsoft Active Directory and LDAP authentication support all are provided. Cloning of the service processor configuration can be done peer-to-peer. In addition, one service processor can be designated as the controller for an entire farm of servers. The management processor also provides zero-footprint diagnostics. Machine check analysis, such as access to memory and processor scans, can be done independently.

Newisys has announced they will supply the core technology integration and packaging engineering expertise, but leave the manufacturing and distribution to external OEMs and licensees. They have contract manufacturing arranged though Sanmina SCI and distribution through Avnet.

Newisys partners include some well-known names in IT: Angstrom Microsystems, APPRO, RackSaver, M&A Technology, Microway, New Technology Solutions, Inc., and ProMicro. At the time of this writing, some 600 systems have been fielded for development and evaluation among OEMs, Fortune 500 companies and, of course, *Linux Journal*.

SuSE Linux Enterprise Server 8 and GNU Applications

Because this is a new fast system, after typing **uname -a** at the bash prompt, the next thing you do is **cat /proc/cpuinfo**:

```
processor   : 0
vendor_id       : AuthenticAMD
cpu family      : 15
model           : 5
model name      : AMD Opteron™
stepping        : 0
cpu MHz         : 1594.286
cache size      : 1024 KB
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr pae mce
    cx8 apic sep mtrr pge mca cmov pat pse36 clflush
    mmx fxsr sse sse2 syscall nx mmxext lm 3dnowext
    3dnow
bogomips        : 3178.49
TLB size        : 1088 4K pages
clflush size    : 64
address sizes   : 40 bits physical, 48 bits virtual
power management: ts ttp
```

Processor 1's information is the same but with a BogoMips of 3185.04.

The main GUI environment supported on SuSE Linux Enterprise Server 8 is KDE 3.0, and the system comes with all the regular tools and applications usually expected with a SuSE distribution. The SuSE Enterprise Linux implementation on the Opteron itself is the result of a lot of hard work and a long history of 64-

bit Linux hacking, dating back to when Jon “maddog” Hall first gave Linus Torvalds a Digital Alpha.

The GNU environment simply works—only at 64 bits. All of the usual compilers and the GNU build environment work right out of the box. One of the first things you notice with this system is that Emacs starts immediately, faster than you can blink.

The first test was to compile and run the GNU Scientific Library (GSL), which is a numerical computing library. This library itself is rather robust, and we compiled version 1.3. Then we compiled some sample code from the GNU Scientific Library Reference Manual to generate random numbers. This really simple program, which was compiled without any optimization, revealed that this system was about three times faster than the same program on x86 at 800MHz.

However, we learned right away how often programming is done with the unconscious thought that it's a 32-bit system. One part of the test program left-shifted an integer 1 by sizeof(int) * 8 - 1 to obtain the number of bits, based on the bit size of the machine's native integer. On this machine, in contrast to typical x86 PCs, compiling the program with that nonportable hack caused immediate integer overflow. Although common tools are now 64-bit clean, your locally developed code may need some work.

We performed the next test with the Icarus Verilog compiler. Icarus Verilog is a GPLed Verilog HDL compiler for electronic design automation (EDA), especially HDL simulation and synthesis. *Linux Journal* has interviewed Stephen Williams, the author of Icarus, two times now [see “Open Source in Electronic Design Automation”, *LJ*, February 2001, www.linuxjournal.com/article/4428 and “A Conversation with Stephen Williams”, *LJ*, July 2002, www.linuxjournal.com/article/6002]. Over time, we have documented the steadily increasing advances this compiler has made as a serious industrial-strength EDA tool for logic and FPGA design. Because Stephen has been building Icarus as 64-bit clean code for the past five years under Linux for Alpha, compiling the compiler was relatively easy. What was of interest was learning how fast it was and how well it responded to large workloads and many more cycles of operation in terms of runtime.

We tested a large multiplier logic model and testbench from the Icarus test suite, and it was almost twice as fast as a 1.5GHz Athlon processor running the same binary. For the large workload test, we compiled a logic model consisting of 1,720,648 lines of Verilog code. This also was breathtaking. In 61 minutes, the machine compiled a model with a memory footprint larger than the largest user space in 32-bit Linux—3.6GB.

Because Icarus uses its own internal threading system, these particular EDA tests used only one-half of this computer. Another processor, with scalable memory bandwidth, runs more EDA simulation projects without even affecting the first. Opteron clearly is well suited as server technology for engineering work.

The last major test compiled was for the Linux Test Project (LTP). The LTP is a GPLed test environment for Linux, available for download on SourceForge (see Resources). The version tested was ltp-full-20030404, which is from early April 2003. It compiled and ran the default tests with no problem whatsoever.

A qualitative summary of the LTP tests would be that 64-bit SuSE Enterprise Linux is squeaky clean—numerous picayune Linux system calls are tested in LTP. There were extremely few failures, and of these, none were of any substantive impact. In our experience, this same LTP on some other recent 32-bit Linux distributions has many more test fails, including actual crashes.

Although the LTP is well geared for testing Linux, some other more quantitative data about the Opteron is necessary to begin to glean numerically the limits to performance for this processor and system architecture. We turn now to some preliminary benchmarks.

Performance Benchmarks

These processors are wicked fast, and it's still early in the system characterization cycle. For example, new applications need to be compiled for a 64-bit operating system to attain the full measure of the available performance. For now, some early industry-standard benchmarks available at press time are listed in Tables 1 and 2. These are for a dual-processor SMP Opteron 244 with PC2700 memory configuration, for 32-bit applications. In Table 1, the benchmarks infer relative integer and floating-point processor performance. The benchmark in Table 2 is designed to evaluate the performance of a server's ability to run Java applications.

Your intended application determines the best benchmark. Programs can appear similar but have unique peculiarities that are stressed differently under system loads.

Table 1. Opteron 244 Benchmarks

Table 2. Opteron 244 Java Middleware Benchmark

Conclusion

The Opteron is breakthrough 64-bit processor technology, one that seems destined to provide high performance and cost savings for cleanly migrating 32-bit architecture programs into a 64-bit application space. In the near future, the upside for 64-bit “long mode” applications running on Linux seems high indeed, because eight-way SMP processors are coming. *Linux Journal* will be covering this emergent technology with additional articles in the future.

Resources

email: mab@cruzio.com

Michael Baxter is technical editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Network Management with Nagios

Richard C. Harlan

Issue #111, July 2003

John Deere had to expand network management across a diverse collection of hardware and software. Nagios saved the day.

When I started with the John Deere Agricultural Marketing Center, we had 12 sites around the US and Canada. These locations had a mixture of equipment, older servers and desktops providing all functions from domain services to printing. That presented a problem to us functioning as a centralized IT organization. How could we manage and monitor the servers and WAN lines in these diverse sites?

About two years ago, we decided to move the user data from the center of our network out to the edges. But how could we monitor all the equipment from multiple manufacturers across all the sites? To give you an idea of the equipment we were monitoring, each of our main sites had a Maxtor 4100 network-attached storage (NAS) server along with a Compaq 1600 serving as a print server and domain controller. Some of the small sites had Dell GX1 desktops serving as local print servers. At our main site, a Compaq TaskSmart N2400 was our main file server.

Because of this diverse setup, no manufacturer's management toolset served our needs, and the idea of using multiple tools to monitor everything was not something I wanted to do. So, we had to choose among manufacturer-neutral monitoring tools. What we found seemed to be grouped into three different tiers. At the low end was a solution that did not expand well to monitor a growing number of servers and hosts. For the most part, we would be stuck with the tools that came with the program. Other tools were what I call top-of-the-line products, such as Hewlett-Packard's OpenView. It would expand and grow with what we needed and monitor what we wanted, but it had a price tag well out of our range.

As we started to lose faith that anything existed to fit our situation, I ran across a project called NetSaint. NetSaint was a monitoring system that would grow with our needs and allow us to monitor what we wanted with an open framework allowing us to write our own plugins. Not only would NetSaint monitor servers and services, it enabled us to make our jobs more proactive instead of reactive by following the trends it showed us.

Everything went along fine, with NetSaint performing its job reliably in one of the smaller units of John Deere. As time went on, the Agricultural Division started an IT divisionalization project, with the goal being the ability to leverage areas of expertise all across the division.

With this new project, the old problem of monitoring resurfaced. As we tried to move to a more centralized IT structure, we needed to perform everything the other monitoring programs did. Also, every unit was on its own to make IT decisions to fit its needs. So the question was: How would we take the multiple monitoring solutions in use and move toward one standard solution? This solution would need be able to handle the possibility of monitoring hundreds of hosts with thousands of services, while allowing individual units to extract data into graphs to monitor trends. Also problematic was an even more diverse set of equipment to monitor, from Network Appliance to Sun to Dell.

We still had the financial constraint, which meant the OpenView class of products was out of our range. Because we were running NetSaint in a production environment, we decided to go ahead and start looking into a NetSaint solution for the Ag Division. The first thing I learned was if you stop following the Open Source community for a while, a lot will have changed when you come back. In this case, NetSaint was no more; its descendant is Nagios (www.nagios.org). For the most part, Nagios is the next step in the NetSaint evolution. I confess to some sadness seeing the penguin removed from the main page, but I got used to it and moved on.

Studying the system, we decided to use a multi-tier model, with Nagios servers at each unit that came onboard with the monitoring project. This was mainly to ensure that no one server would be overloaded, allowing us to maintain the level of monitoring we needed. With this in mind, we installed the parent server at our headquarters in Moline, Illinois, with the first child server in our Ag Marketing location in Lenexa, Kansas.

The install of Nagios was the same for both the parent server and the child server. We loaded Red Hat 8 on both and then installed Nagios. The documentation that comes with the system is fairly complete and walks you through the install process easily. Along with Nagios, we installed a program called Nagat on each of the servers (Figure 1). Nagat is a web-based

configuration tool for Nagios. With this install, the system allows you to fill in the blanks on the web page and generates the configuration files based on your answers. Therefore, less-experienced people can configure Nagios, its services and hosts without touching a terminal window.

Figure 1. The Nagat Services Page

Because we used Nagat, we ran into some difficulties with our Red Hat 8 install. To fix the compatibility problems, we had to install earlier versions of PHP (4.1.2) and Apache (1.3). With this resolved, we were able to get the systems up and running quite quickly. Nagat had a few bugs that were easy to work out, one occurring on the service edit page: the system does not save the contact group. To fix this, all you have to do is add the following lines after the case statement on line 39:

```
$saveobject['contact_groups'] =
    @implode("'",$saveobject['contact_groups']);
```

By doing this, you will be able to save changes to your contact groups when you update your services.

Once everything was installed, we came to what I would call the hardest part of the whole project—the configuration of the servers. It wasn't so much configuring the server as it was figuring out what we wanted our system to do and the path to take to get it done. One of the first decisions we made—one we now are reversing—was to compile Nagios without native database support. It met our needs for over a year, but as we bring more units onto the Nagios monitoring platform, mining the data provided from Nagios is becoming more of a priority. No longer are the simple graphs that come with Nagios enough. We want to take the data and trend it out in as many ways as there are units and people, which was not all that easy to do with the flat text files we had with our original Nagios install.

The next step was the main configuration, which for the most part is really straightforward. There are a few things you need to look into and explore a little to get the most out of the Nagios system. One of them is the idea of smart checks. Instead of the system running all of your service checks at the same time, causing high CPU utilization, Nagios spreads out the checks over, say, five minutes, causing lower CPU utilization. Another thing to look at is the parallelized service checks; they allow you to run more than one service check at a time, which really helps on systems with multiple CPUs.

The configurations of the parent and child servers are almost identical, with two important differences. The child server has notification turned off and the active checks turned on by default, and the parent server has the exact opposite. This is done to lower the load on the parent server and to have all notifications sent out from only one computer. [The configuration files for both servers are available at ftp.linuxjournal.com/pub/lj/listings/issue111/6767.tgz.]

From there, the next step is defining the commands Nagios should perform. This is one thing I stress to the people using this system: Nagios is a framework. What I mean is, Nagios itself does not perform any checks; that work is left to the plugins that Nagios can call. This is good because it allows you to write your own plugins easily, as long as you stick to the framework Nagios provides. As a result of the plugins we have written, we have the ability to monitor systems that were not originally in the Nagios plugin distribution. For example, we now are able to monitor NetApp filers, to pull the page counts from our HP printers and to integrate Compaq Insight Manager data into the Nagios system.

The easy way to monitor a Microsoft Windows server, but not the only way, is to use NSClient. NSClient is a program loaded on your Windows boxes, and it runs as a service. To run it, open up a port on the Windows server. For security, you

can use a password to access the port. At the time of this writing, no encryption is built in to the program for talking to the Nagios server. The plugin allows you to poll the server for memory utilization, disk utilization, processor load and most other information that can be polled using the performance manager tool with Windows.

A slightly different method is used to monitor your Linux and UNIX systems. Run a program called Nagios Service Check Acceptor (NSCA), either in dæmon mode or under inetd. With this program, Nagios starts a plugin called check_nsca, which opens an encrypted session between the two computers and runs the plugin to check the other computer. The configuration of the system is rather straightforward; pick an encryption standard between the two servers and define the plugin command on the client. Once this is done, you can start pulling the data back into Nagios.

As I said earlier, we are now in the process of moving from the flat files that are the default for Nagios to a back-end database. We plan to use that data to provide better data trending through the use of tools such as Crystal Reports. Nagios provides native support for two databases, MySQL and PostgreSQL. Scripts are provided with the Nagios files for setting up the database and preparing all the fields and tables for Nagios. We decided to use PostgreSQL. Once the database is all set up, you need to rerun the configuration script with **--with-mysql-xdata** or **--with-pgsql-xdata**. **xdata** sets up the system to use the database for everything. As of right now, the only pieces that do not support the database are the configuration files.

That last statement brings up an interesting point: an add-on came out in March 2003, a Webmin plugin for Nagios (Figure 2), called NagMIN. Not only does it allow web-based configuration of Nagios, like Nagat, it offers a few things that nothing else has offered so far. First, it offers database support of the Nagios configuration files. Second, something quite useful if you are starting to set up your Nagios install, is the port scan feature (Figure 3). I have not tried out this feature fully yet, but it claims to perform a network discovery and turn what it finds into Nagios configuration files, saving you the trouble of having to enter all the data yourself. NagMIN should work with most systems, but the more-specialized monitoring most likely still has to be entered into the system by hand. If you would like more information about this plugin, go to the SourceForge.net web page.

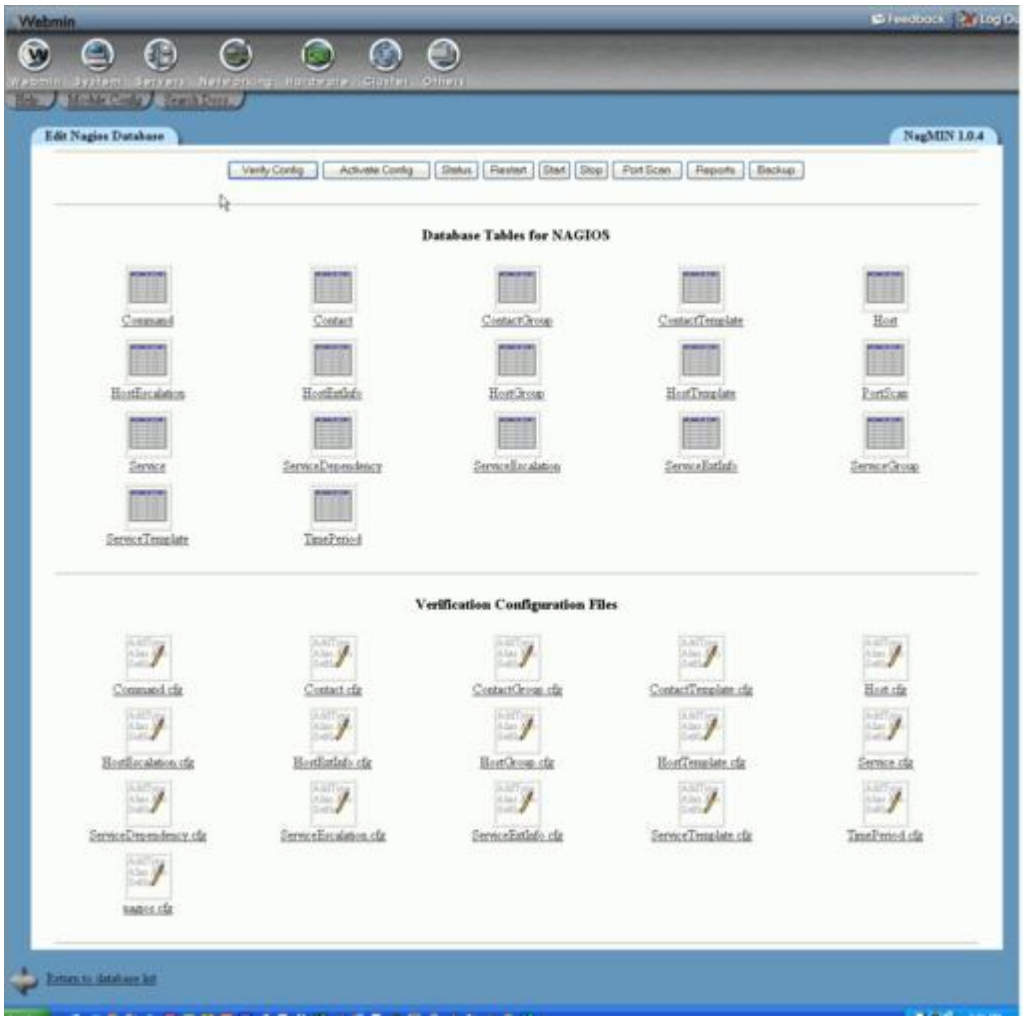


Figure 2. NagMIN Main Page

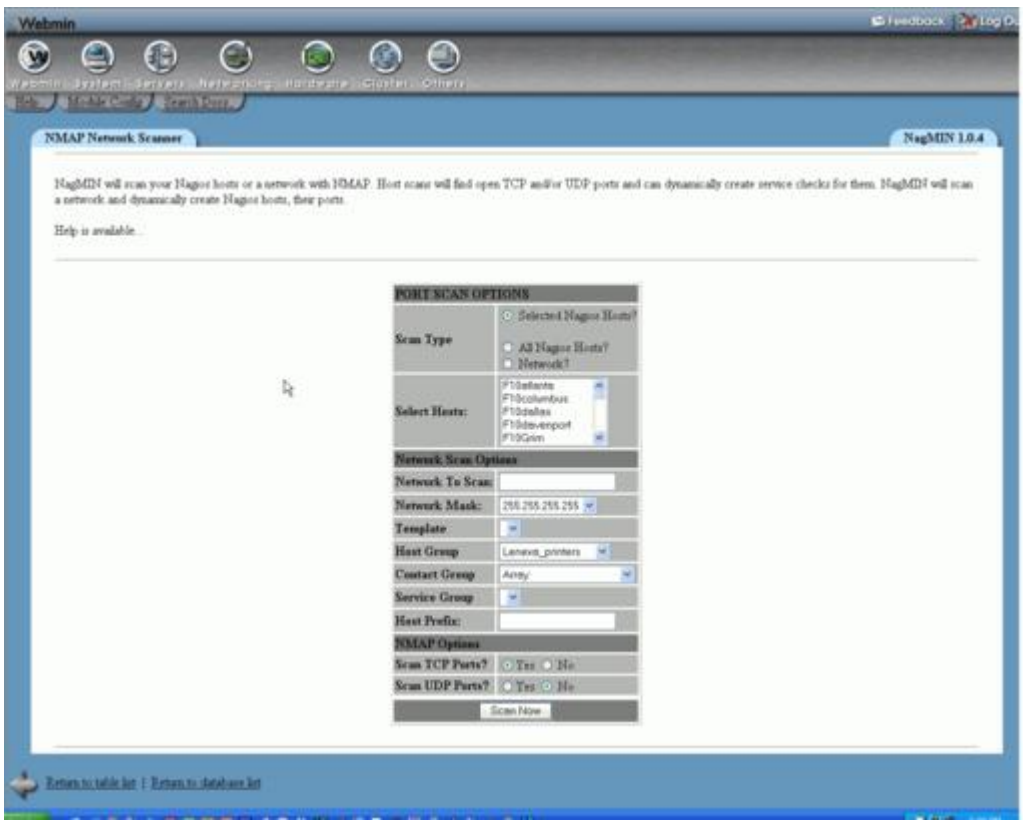


Figure 3. A NagMIN port scan discovers services to monitor.

We set up our Nagios server in a parent/child relationship. To do this, the system needs to be set up a little differently from what it would be for a standalone system. For one thing, we installed only the web interface on the parent server (Figures 4 and 5). All the child servers do is funnel their check results to the parent server using passive checks. To make this work, we defined an OCSP command to run a script to submit the data to the parent Nagios server. By using the OCSP command to run the script, it is made to execute after every check that Nagios runs. Because of this, for the most part, only the child servers need to run active checks. This allows the Nagios parent server to run only the web interface and to send out notifications in case of problems.

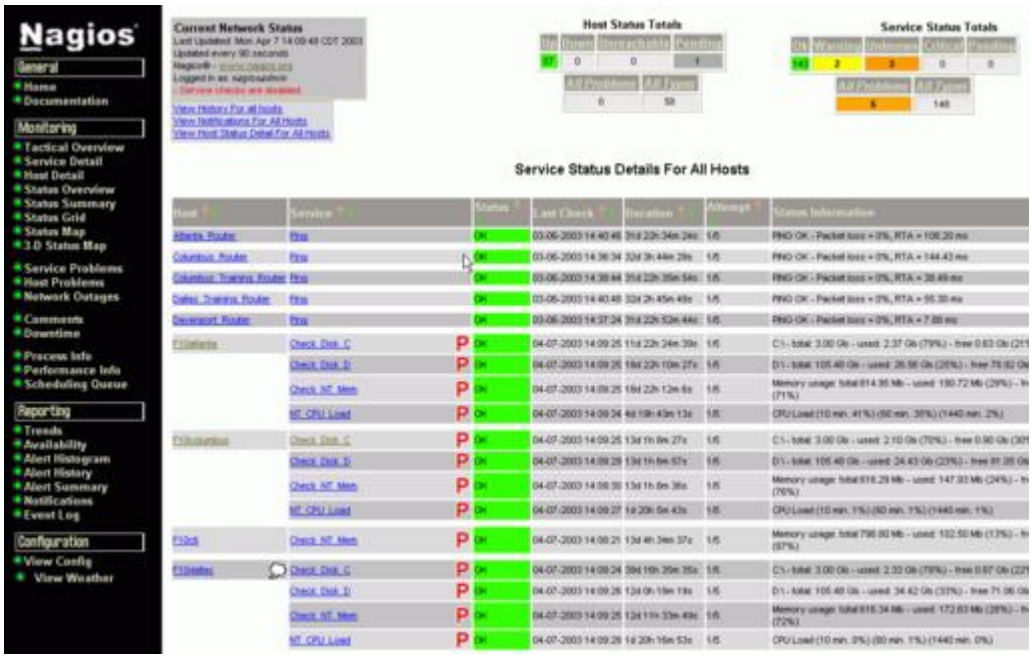


Figure 4. Nagios Service Detail

Figure 5. Status Summary

Now that you have an idea of how we set up Nagios, let me tell you about something it has done for us. Around the first of this year we purchased a digital thermometer, called TempTrax, that works with Nagios. We use it to take temperature readings of our main computer room. Around the first of March, we learned how important Nagios is to us. I received a page at about 12**AM** on a Friday telling me there was a warning on the computer room temperature. In the time it took me to get dressed, the system started sending out critical pages about the computer room temperature. By the time I got to work, the temperature was above 80° and rising. Because of Nagios, I had enough time to make an emergency call to our air-conditioning people and have the unit repaired before any serious damage was done to the computers. We found out later that two other systems monitoring the computer room A/C unit had failed to dial out and alert the people watching the system. If Nagios had not alerted us to the problem in the computer room, I would have walked into a much bigger problem that next morning.

Overall, I would have to say that Nagios is an excellent network monitoring product. As described, it is a framework, and by itself it can't do much—but that is what makes it such a good product. Because it is a framework, it does only what you want it do. Being an open framework, designing new plugins with Nagios is as simple as formatting the output from your check into the format that Nagios expects. With this your plugin is able to use everything Nagios has to offer, which is quite a lot. In the end, Nagios is something you should at least take a look at, if not test install at your own location. I do not think you will be disappointed.

email: HarlanRichard@JohnDeere.com

Richard C. Harlan is a network engineer at the John Deere Ag Marketing Center in Lenexa, Kansas. You can reach him at harlanrichardc@johndeere.com.

Archive Index Issue Table of Contents

Advanced search

Getting to Know Mono

Julio David Quintana

Issue #111, July 2003

Mono is useful for more than simply getting Linux to work with Microsoft's .NET. It offers you a chance to use libraries in one language from another without writing a wrapper.

If you have ever written an application for the Linux desktop, or even looked into writing one, you are familiar with the multitude of language bindings available for the various GUI toolkits. This is one of the strengths of writing GUI applications for Linux; you are not locked in to a particular programming language. Unfortunately, you quickly come to realize that different language bindings offer varying amounts of API completeness. A widget you used from one language isn't yet supported when using a different language. This is the downside of supporting multiple languages. The amount of work needed to maintain an API increases with each set of bindings. A change or update to the original API must be replicated in each of the language wrappers.

Now imagine a single GUI toolkit, accessible from any programming language without having to rely on API wrappers—a toolkit that offers the same functionality to every language that uses it. Mono has the potential to provide this, plus much more, by offering programming language independence as well as programming language interaction.

Brief History

Life for Mono began about two years ago at the Linux software company Ximian, Inc. Ximian is known for their Ximian Desktop, Evolution PIM/e-mail client, Red Carpet upgrade system and enthusiastic CTO Miguel de Icaza. Recognizing the potential in a couple of newly proposed standards, Miguel de Icaza began prototyping what would later become the Mono Project.

So what were these standards that caught Miguel de Icaza's eye? It's no secret that they were ECMA-334 and ECMA-335, the specifications for the core

technologies in Microsoft's .NET development platform. At this point, it probably is important to point out that there is a difference between the .NET development platform and the blanket term ".NET". Microsoft covers a whole slew of products and services, including operating systems, development tools, network services and applications, with the expansive .NET term. We are concerned with only a portion of .NET.

In October 2000, Microsoft, Hewlett-Packard and Intel jointly submitted the specifications for a runtime environment known as the Common Language Infrastructure (CLI) and a newly developed object-oriented language named C#. By the second half of 2001, Ximian officially had launched the Mono Project to provide an open-source implementation of the .NET development platform based on the proposed standards. In December 2001, the European Computer Manufacturing Association (ECMA) officially ratified as standards the specifications for the CLI and C# language.

Sidebar: Brief Introduction to C#

Overview

The CLI lays out a base class library and a runtime environment that provides services such as Just In Time (JIT) compilation, memory management, exception handling, loading and linking and security management. To illustrate this better, it helps to compare it to the traditional method of compiling source code.

Traditionally, source code is converted by the compiler to machine-specific instructions. The instructions are then executed directly on the processor. A program compiled for the x86 processor line will fail to execute on a PPC processor without first being recompiled for that processor. This makes it difficult for software to target multiple hardware platforms, as a different version of compiled code must be kept for each one.

As an alternative, source compiled for a runtime environment is converted to an intermediate set of instructions that are not dependent on the underlying hardware. The intermediate instructions then can be executed in a couple of different ways. One method is to use a interpreter. The interpreter loads the intermediate instructions and then executes them, in essence acting as a virtual machine. In a second method, the intermediate form is JIT-compiled at runtime or installation time into machine-specific instructions and then executed directly. Because JIT compiling executes native platform instructions, compiling can be optimized for the target processor. The JIT compiler can increase execution speed further by converting only the portions that are being used into native instructions and then storing those in memory for subsequent calls.

The trade-off for having the platform independence of using a runtime environment is in execution speed. Compared to the traditional method of compiling to native instructions, the runtime is slower. How much slower depends on the specific situation and which method of execution the runtime uses. Generally though, an interpreter provides the slowest execution speed. The performance of a JIT compiler is much closer to the performance of traditional compiling because both produce native instructions. The overhead of the runtime itself still keeps the speed performance slightly behind.

I know what you are thinking. An object-oriented language, a base class library, a runtime environment—this sounds a lot like Java. Well, you are right. The components of the CLI are very similar to those found in Java. However, there is one fundamental difference. The Java runtime was designed only for the Java language. Although it is true that a handful of other languages have been ported to output Java bytecode and run on the JVM, this still falls short of the language neutrality supported by the CLI. From the ground up, the CLI was designed to be the execution environment for many programming languages. The data type system of the CLI can support imperative languages, like C or Pascal, as well as object-oriented languages. Not only does the CLI have facilities to execute multiple languages (language independence), it also provides the framework to allow those languages to share data with each other (language interaction), including cross-language exception handling. An object created in one language can be inherited in another. Details of how the CLI achieves this level of language neutrality can be found by examining its core components.

Common Type System

At the heart of the CLI is the Common Type System (CTS). The CTS defines a shared data type system and the rules used in declaring, using and managing the types. By employing a strictly enforced type system, the CLI can ensure that type safety is maintained as well as make it possible for languages to interoperate with types of another language. In order to accommodate a multitude of different programming languages, the CTS furnishes two main data types that contain multiple subtypes, values (value types) and objects (reference types). Values are reserved for representing simple data types such as integers and floating-point values. Objects are used for the more complex entities required by programming languages.

Common Language Specification

The Common Language Specification (CLS) outlines the framework compilers must adhere to when generating libraries and binaries for cross-language interaction. The CLS is actually a subset of the CTS, providing a reasonable type system and rules a language compiler must support in order to produce

compiled code that can be used or extended by other languages. A language has the ability to choose how much of the CLS to support. Languages that allow any CLS type to be used are called CLS consumers. Languages that allow CLS types to be created or extended are called CLS extenders. A language that fully embraces the CLS is both a consumer and extender.

Metadata and the Common Intermediate Language

When a source file is compiled by a CLI-compliant compiler, a binary file called a portable executable (PE) or sometimes referred to as an assembly, though an assembly can consist of one or more files, is output. The PE contains two important pieces of information. The first is metadata. Metadata is used to describe the types used as well as information the CLI uses to locate and load classes, lay out memory and other execution-time information. The second piece is the Common Intermediate Language (CIL) bytecode. CIL is a language-independent set of intermediate instructions. When a language is compiled for the CLI, CIL bytecode is produced. CIL is robust enough to to handle a myriad of different programming languages and is designed to be converted efficiently to native platform instructions. A snippet of CIL instructions for a “hello world” program written in C# can be seen in Listing 1.

Listing 1. Part of a Disassembled C# Program Showing the CIL Instruction Set

Virtual Execution System

The Virtual Execution System (VES) provides the environment for executing programs written for the CLI. It loads, links, manages memory, handles security and exceptions and provides the support framework for executing CIL instructions.

The memory management supplied by the CLI is administered by a garbage collector (GC). Unlike other runtime environments, the GC of the CLI can be switched on and off within the source code. The data allocated and destroyed by the GC is called managed data. When the GC is not used on data, it is referred to as unmanaged data. Managed code, source code executed by the CLI, can access both managed and unmanaged data.

Mono

The goal of the Mono Project is twofold. First, it provides an implementation of the ECMA standards for the CLI and C#. Second, it adds compatibility with the Microsoft .NET development platform. Each part has its own intrinsic value and benefits Linux in different ways. If, for example, .NET compatibility was no longer available, Mono would remain a valuable development framework for Linux.

However, by adding .NET compatibility to the Linux platform, Mono makes software developed for Windows available to Linux. Along the same line of thought, developers wishing to make the transition to Linux application development will have access to a development framework with which they are already familiar, thereby lowering the learning-curve barrier.

The major portions of .NET that the Mono Project is working on delivering are Win Forms (System.Windows.Forms), ADO.NET and ASP.NET. Win Forms contain all the necessary methods, classes and events for creating GUI applications compatible with Microsoft Windows. Because it is nearly impossible to emulate the Windows GUI API calls with native Linux GUI toolkits, Mono is using WineLib (www.winehq.com) to provide the Windows interface. If you have ever seen an application running in Wine, you know it looks nothing like Linux desktop environments. To solve this, Mono is looking to add theming support to Wine to use the same rendering routines for the widgets as the rest of the desktop.

ADO.NET contains the .NET data access classes for Mono. ADO.NET offers more than mere database access. It provides a model for accessing data from any source in a disconnected, scalable method based on XML. At the time of this writing, about a dozen databases are working as Mono ADO.NET data providers. Work continues to increase the maturity and add additional database vendor support.

ASP.NET support in Mono is divided into two parts, web forms and web services. Web forms create the user interface for a web application. Much like Win Forms, web forms provide properties, methods and events for controls such as buttons, text boxes or complex controls made of multiple simple controls. This allows web form interfaces to be created in rapid application development (RAD) environments using drag-and-drop techniques similar to Glade on GNOME. This separates the presentation from the logic and lessens the amount of coding needed. Web services offer SOAP-based remote procedure call support. Using ubiquitous internet protocols like XML and HTTP, web services allow the sharing of data or logic over the network and even through firewalls. Any language supported by the CLI can be used to program the logic for ASP.NET. This also means that ASP.NET code is compiled and not interpreted like previous versions of ASP and other web-scripting languages. ASP.NET is available for Mono in either the XSP web server or in the mod_mono component for Apache 2.

In addition to the Mono class libraries implementing .NET, several other libraries and tools offer interesting and useful functionality:

- GTK#, Qt# and Wx.NET provide C# bindings for the popular Linux GUI toolkits. With these C# wrappers, all languages that can run on Mono have access to the same GUI toolkits:
- OpenGL#, MonoGLo and CsGL provide bindings for the popular 2-D/3-D graphics API OpenGL.
- SDL.NET provides bindings for the SDL game library.
- Gst# Gstreamer multimedia framework bindings.
- Many communication libraries, including .NET Jabber and Gnutella.
- NAnt build tool (similar to Ant).

Of course, these are only a few examples, but they're enough to illustrate the potential for developing with Mono for Linux and other platforms.

Using Mono

The first step in taking Mono out for a test spin is to visit the project web site (www.go-mono.com) and download the latest source tarballs or platform binaries. Currently, Mono has been ported only to Linux and Windows, but work is being done on Mac OS X, FreeBSD and other platforms. Binaries are available for a variety of Linux distributions including Debian, Red Hat, SuSE and Mandrake. If you use Ximian Red Carpet, the files also are available in the Mono Channel. For this article, we are using Mono version 0.20. You'll notice that in addition to the Mono packages providing the runtime, C# compiler and class libraries, there are a few other goodies to play with such as the Mono debugger, XSP web server and Monodoc documentation browser.

If you have trouble installing Mono, check out the tutorials offered on the web site.

Mono currently comes packaged with the following components:

- C# and Basic language compilers.
- VES consisting of a JIT compiler and associated garbage collector, security system, class loader, verifier and threading system. An interpreter is also included.
- A set of class libraries written in C# that implements the classes defined in the CLI standard, classes that are part of the .NET FCL, and other Mono-specific classes.
- Various utilities.

The Mono C# language compiler is mcs. In an interesting programming feat, mcs is written in C#. Since Mono 0.10, mcs even has been able to compile itself. If you are interested in the details of the command-line options, which are compatible with the command-line options provided by Microsoft's C# compiler, a thorough man page is available.

The compiler for Mono's equivalent of Visual Basic.NET, MonoBasic, is mbas. Although not as far in development as the C# compiler, mbas provides enough functionality to experiment a little in Basic.

Two execution environments are included with Mono, mono and mint. mono is a JIT compiler compatible with the CLI's definition of the VES. mint on the other hand, is an interpreter. It is provided as an easy-to-port alternative to mono, which currently runs only on the x86 platform. For the greatest code execution speed, use mono.

A couple interesting utilities also provided with Mono are monodis and pedump. monodis is used to disassemble a compiled assembly and output the corresponding CIL code. It was used to display the sample CIL code for Listing 1. If you are curious to see more of what CIL looks like or to take a peek at what makes up a portable executable, play around with these.

Now that we are familiar with the components of Mono, it is time to try them out. To experiment with the language interaction of Mono, we write a simple class with a single method in C# and call it from a MonoBasic program.

Listing 2 shows the C# library ljlib.cs, and Listing 3 shows the MonoBasic program hello.vb.

Listing 2. ljlib.cs

Listing 3. hello.vb

The first step is to compile the ljlib.cs into a library. Compiled libraries have the .dll extension, and compiled executables have the .exe extension. To compile to a library, use the -target:library switch in mcs:

```
[jdq@newton]$ mcs -target:library ljlib.cs
Compilation succeeded
```

This creates the ljlib.dll file, which contains the *LJ*lib namespace and Output class. Now we need to compile the hello.vb program. In order to use the ljlib.dll file we just created, we need to tell the MonoBasic compiler to use it as a reference. We do that with the -r switch:

```
[jdq@newton]$ mbas -r ./ljlib.dll hello.vb
Compilation succeeded
```

The output of mbas is the PE hello.exe. It can be executed with mono:

```
[jdq@newton]$ mono hello.exe
Hello Linux Journal!
```

And there you have it—two languages, C# and MonoBasic, executing on the same runtime and working together. This is a trivial example; however, it does demonstrate the language independence and interoperability of the CLI and hints at the power of Mono as a development platform.

Conclusion

Though still in development, Mono shows promise for greatly benefiting Linux development. With the progress of the last two years as a gauge, the future of Mono should prove to be quite exciting.

Resources

email: david@davidquintana.com

Julio David Quintana is an electrical engineer who stumbled upon Linux in 1997 and has been hooked ever since. If you find grammatical or factual errors, he is unable to be contacted. However, for praises and compliments, jdq@jdqi.com works just fine.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

How to Index Anything

Josh Rabinowitz

Issue #111, July 2003

You probably have search on your web site, but how about a search engine for the man pages on your system or even your mail? Try this simple indexing package.

You might want to build custom indices of documents for many reasons. A widely cited one is to supply search functionality to a web site, but you also may want to index your e-mail or technical documents. Anyone who has looked into implementing such a functionality has probably found it's not as easy as it might seem. Various factors conspire to make searching difficult.

The venerable and indispensable grep and its ilk are effective for scanning through lines of text. But grep, egrep and their relations won't do everything for you. They won't search across lines, they won't show search results in a ranked order and their linear search algorithms don't lend themselves to searching larger volumes of data.

HTML doesn't help the situation either. Its display-oriented features, idiosyncratic grammar and bevy of formatting and entity tags make it fairly difficult to parse correctly.

At the other end of the data storage spectrum is data slotted into a database. The ubiquitous example is that of the SQL database, which allows somewhat sophisticated search facilities but usually is not particularly fast for searching. Some database engines, notably MySQL 4, address this issue by allowing fast and ranked searches, but they may not be as customizable as desired.

In this article, we explore ways to create custom indices using SWISH-E, Perl and XML on Linux. Through examples, we show how SWISH-E can be used to build indices of HTML files, PDF files and man pages.

SWISH-E (simple web indexing system for humans—enhanced) is a descendant of SWISH, which was created in 1994 by Kevin Hughes. SWISH was transferred in 1996 to the UC Berkeley Library to fix bugs and add features, and the result was licensed under the GPL and renamed SWISH-E. Development continues, spearheaded by current project maintainer Bill Moseley and assisted by a team of developers.

Here at SkateboardDirectory.com, we happened upon SWISH-E when researching indexing toolkits. We found that it offers a unique combination of features that make it attractive for our purposes. Not only does SWISH-E offer a fast and robust toolkit with which to build and query indices, but it is also well documented, undergoes active development and bug fixes and includes a Perl interface. We also liked that maintainer Moseley and other experienced SWISH-E users and developers are usually prompt when addressing questions and bugs brought up on the SWISH-E mailing list.

Installing SWISH-E

For our examples, we started with a stock Red Hat 7.3 workstation with the Software Development bundle of packages installed. We also tested the examples on a Red Hat 6.2 workstation and a Debian Woody.

Currently, installing SWISH-E on Red Hat means installing from source, and the zlib and libxml2 libraries are required to build SWISH-E fully. If you find you need to install either, you probably can find packages provided with your distribution. We also use the xpdf package in our examples, so you may want to install that now if it isn't already. Our reference Red Hat 7.3 workstation setup had all of SWISH-E's prerequisites installed.

Here, we describe the use of SWISH-E 2.4, which according to the development team, should be released by the time you read this article. You can fetch and set up SWISH-E with the following sequence of commands, substituting the current version for (*x.x*):

```
% wget \
  http://swish-e.org/Download/swish-e-x.x.tar.gz
% tar zxf swish-e-x.x.tar.gz
% cd swish-e-x.x
% ./configure
% make
% make test
```

To install the SWISH-E binary, C libraries and man pages into their default locations in /usr/local, type **make install** as root. This installs the SWISH-E executable into /usr/local/bin. If this directory isn't in your PATH, either change your appropriate dot file to include /usr/local/bin in your PATH, or always call the swish-e executable by full pathname, like /usr/local/bin/swish-e.

Now, let's build and install the SWISH::API Perl module from the Perl directory in the source. We'll need it later when we build a Perl client for our index of man pages. SWISH::API is set up by the normal Perl module install process:

```
% cd perl
% perl Makefile.PL
% make
% make test
```

Then, install the SWISH-E Perl module by typing **make install** as root.

Now that SWISH-E and the SWISH::API Perl module are installed fully, let's build a simple index of HTML files to test SWISH-E. For this example, we index the HTML, one-page-per-section versions of the Linux Documentation Project (LDP) HOWTOs, which we've unpacked into ~/HOWTO-htmls/. The tarballs of LDP documents used in this article come from www.tldp.org/docs.html.

Indexing HTML on the Filesystem

The first step in building an index with SWISH-E is writing a configuration file. Create a directory like ~/indices, **cd** into it and create the file ./howto-html.conf with the following contents:

```
# howto-html.conf
IndexDir  ../HOWTO-htmls/
IndexOnly .html
IndexFile ./howto-html.index
```

The IndexDir directive specifies the directory in which SWISH-E should look for files to be indexed. The IndexOnly directive requests that only files ending in .html be indexed. Finally, the location of the index to be created is specified with the IndexFile directive.

Our First Index

Now, let's build our index of HTML files with the command:

```
% swish-e -c howto-html.conf
```

The -c option specifies which SWISH-E configuration file to use. On an older system, building this index may take a few minutes or so; on a contemporary one, it should take under a minute. Figure 1 illustrates the process of indexing HTML files on the filesystem with SWISH-E.

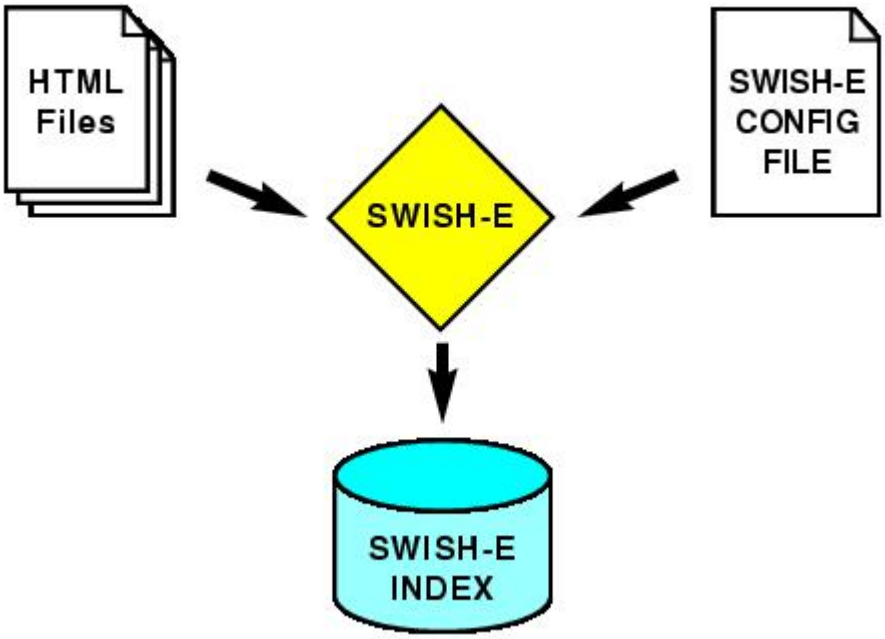


Figure 1. Indexing HTML on the Filesystem with SWISH-E

Searching the Index

Let's test our first index by doing a simple search for HTML files relevant to the term NFS. You can test SWISH-E indices quickly using the swish-e executable by specifying an index with the -f option, and the text to be searched with the -w option; searches on SWISH-E indices are case-insensitive. Because we expect a lot of pages (or hits) to include the word NFS, we use the -m 3 option to request only three:

```
% swish-e -f howto-html.index -m 3 -w nfs
```

This returns (abridged and reformatted):

```
1000 ../HOWTO-htmls/NFS-HOWTO/performance.html
        "Optimizing NFS Performance" 33288
998 ../HOWTO-htmls/NFS-HOWTO/intro.html
       "Introduction" 10966
993 ../HOWTO-htmls/NFS-HOWTO/security.html
       "Security and NFS" 35968
```

Not bad—those pages are definitely about NFS, and the output is intuitive. The first column is the rank SWISH-E gives each hit—the hits considered most relevant always are ranked 1000, with less-relevant files ranked in descending order. The second column shows the name of the file, the third gives the page's title and the fourth shows the byte count of the indexed data. SWISH-E determines the title of each page from the HTML tags in each file using one of its HTML parsing engines.

The built-in SWISH-E parsing engines are called TXT, HTML and XML, and each is designed to parse the corresponding type of content. Recent versions of

SWISH-E also can use the libxml2 library for the HTML2 and XML2 parsing back ends. Both the XML2 and HTML2 parsers are preferable to their built-in counterparts—especially HTML2. This is why a recent version of libxml2, though technically optional when building SWISH-E, probably should be considered a prerequisite.

Basic SWISH-E Search Syntax

SWISH-E supports a full-featured text retrieval search language with syntax including AND, OR, NOT and parenthetic grouping that all work predictably. For example, the following searches all have the expected semantics:

```
% swish-e -f howto-html.index -w nfs AND tcp
% swish-e -f howto-html.index -w nfs OR tcp
% swish-e -f howto-html.index \
        -w '(gandalf OR frodo) OR (lord AND rings)'
```

The Configuration File

SWISH-E configuration files are simple text files in which each line is either a directive or a comment. Any line in which the first non-whitespace character is a # is ignored by SWISH-E as a comment. All other non-empty lines should be in the form:

```
Directive Options [Options] ...
```

If you need to specify an option with spaces embedded, you can use quotation marks:

```
Directive "Option With Spaces!"
```

If the option has single quotation marks within it, you can quote it with the double quote character and vice versa, for example:

```
Directive "Fred's Index Option"
Directive 'By Josh "joshr" Rabinowitz'
```

Dozens of directives can be applied to SWISH-E configuration files. An exhaustive reference can be found in the SWISH-E documentation.

The Index

Each SWISH-E index is stored in a pair of files. One is named as specified in the IndexFile directive, and the other is called *indexname*.prop. When talking about a SWISH-E index, we mean this pair of files.

The indices can get large. In our example index of HTML files, the index occupies about 11MB, about one-fourth the size of the original files indexed.

Indexing PDF Files

Up to now, we've talked only about indexing HTML, XML and text files. Here's a more-advanced example: indexing PDF documents from the Linux Documentation Project.

For SWISH-E to index arbitrary files, PDF or otherwise, we must convert the files to text, ideally resembling HTML or XML, and arrange to have SWISH-E index the results.

We could index the PDF files by converting each to a corresponding file on disk and then index those, but instead we'll use this opportunity to introduce a more flexible way to index data: SWISH-E's programmatic access method (Figure 2).

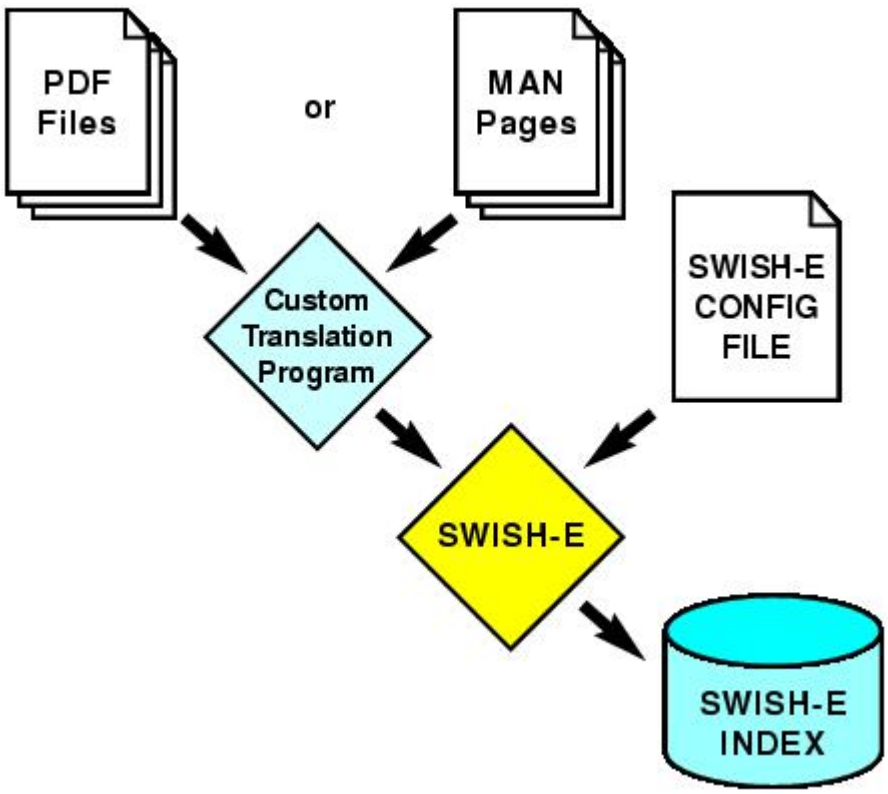


Figure 2. Indexing Arbitrary Data with an External Program and SWISH-E

To index the PDF files, start by creating a SWISH-E configuration file, calling it howto-pdf.conf and endowing it with the following contents:

```
# howto-pdf.conf
IndexDir      ./howto-pdf-prog.pl
               # prog file to hand us XML docs
IndexFile     ./howto-pdf.index
               # Index to create.
UseStemming   yes
MetaNames     swishtitle swishdocpath
```

Here, the IndexDir directive specifies what SWISH-E calls an external program that will return data about what is to be indexed, instead of a directory containing all the files. The UseStemming yes directive requests SWISH-E to stem words to their root forms before indexing and searching. Without stemming, searching for the word “runs” on a document containing the word “running” will not match. With stemming, SWISH-E recognizes that “runs” and “running” both have the same root, or stem word, and finds the document relevant.

Last in our configuration file, but certainly not least, is the MetaNames directive. This line adds a special ability to our index—the ability to search on only the titles or filenames of the files.

Now, let's write the external program to return information about the PDF files we're indexing. Conveniently, the SWISH-E source ships with an example module, pdf2xml.pm, which uses the xpdf package to convert PDF to XML, prefixed with appropriate headers for SWISH-E. We use this module, copied to ~/indices, in our external program howto-pdf-prog.pl:

```
#!/usr/bin/perl -w
use pdf2xml;
my @files =
    `find ../HOWTO-pdfs/ -name '*.pdf' -print`;
for (@files) {
    chomp();
    my $xml_record_ref = pdf2xml($_);
    # this is one XML file with a SWISH-E header
    print $$xml_record_ref;
}
```

Equipped with the SWISH-E configuration file and the external program above, let's build the index:

```
% swish-e -c howto-pdf.conf -S prog
```

The -S prog option tells SWISH-E to consider the IndexDir specified as a program that returns information about the data to be indexed. If you forget to include -S prog when using an external program with SWISH-E, you'll be indexing the external program itself, not the documents it describes.

When the PDF index is built, we can perform searches:

```
% swish-e -f howto-pdf.index -m 2 -w boot disk
```

We should get results similar to:

```
1000 ../HOWTO-pdfs/Bootdisk-HOWTO.pdf
      "Bootdisk-HOWTO.pdf" 127194
983 ../HOWTO-pdfs/Large-Disk-HOWTO.pdf
     "Large-Disk-HOWTO.pdf" 85280
```

The MetaNames directive also lets us search on the titles and paths of the PDF files:

```
% swish-e -f howto-pdf.index -w swishtitle=apache
% swish-e -f howto-pdf.index -w swishdocpath=linux
```

All corresponding combinations of searches are supported. For example:

```
% swish-e -f howto-pdf.index -w '(larry and wall)
    OR (swishdocpath=linux OR swishtitle=kernel)'
```

The quoting above is necessary to protect the parentheses from interpretation by the shell.

Indexing Man Pages

For our final example, we show how to make a useful and powerful index of man pages and how to use the SWISH::API Perl module to write a searching client for the index. Again, first write the configuration file:

```
# sman-index.conf
IndexFile ./sman.index
  # Index to create.
IndexDir  ./sman-index-prog.pl
IndexComments no
  # don't index text in comments
UseStemming yes
MetaNames     swishtitle desc sec
PropertyNames            desc sec
```

We've described most of these directives already, but we're defining some new MetaNames and introducing something called PropertyNames.

In a nutshell, MetaNames are what SWISH-E actually searches on. The default MetaName is swishdefault, and that's what is searched on when no MetaName is specified in a query. PropertyNames are fields that can be returned describing hits.

SWISH-E results normally are returned with several Auto Properties including swishtitle, swishdesc, swishrank and swishdocpath. The MetaNames directive in our configuration specifies that we want to be able to search independently not only on each whole document, but also on only the title, the description or the section. The PropertyNames line specifies that we want the sec and desc properties, the man page's section and short description, to be returned separately with each hit.

The work of converting the man pages to XML and wrapping it in headers for SWISH-E is performed in Listing 1 (sman-index-prog.pl).

Listing 1. sman-index-prog.pl converts man pages to XML for indexing.

```
#!/usr/bin/perl -w

use strict;
use File::Find;

my ($cnt, @files) = (0, get_man_files());
warn scalar @files, " man pages to index...\n";
for my $f (@files) {
    warn "processing $cnt\n" unless ++$cnt % 20;
    my ($hashref) = parse_man($f);
    my $xml = make_xml($hashref);
    my $size = length $xml; # NOTE: Fails if UTF
    print "Path-Name: $f\n",
       "Document-Type: XML*\n",
       "Content-Length: $size\n\n", $xml;
}

sub get_man_files {  # get english manfiles
     my @files;
     chomp(my $man_path = $ENV{MANPATH} ||
       `manpath` || '/usr/share/man');
     find( sub {
       my $n = $File::Find::name;
       push @files, $n
       if -f $n && $n =~ m!man/man.*\.!
    }, split /:/, $man_path );
    return @files;
}
sub make_xml { # output xml version of hash
    my ($metas) = @_; # escapes vals as side-effect
    my $xml = join ("\n",
    map { "<$_>" . escape($metas->{$_}) .
"</$_>" }
    keys %$metas);
    my $pre = qq{<?xml version="1.0"?>\n};
    return qq{$pre<all>$xml</all>\n};
}
sub escape { # modifies scalar you pass!
    return "" unless defined($_[0]);
    s/&/&amp;/g, s/</&lt;/g, s/>/&gt;/g for $_[0];
    return $_[0];
}

sub parse_man {   # this is the bulk
    my ($file) = @_;
    my ($manpage, $cur_content) = ('', '');
    my ($cur_section,%h) = qw(NOSECTION);
    open FH, "man $file  | col -b |"
    or die "Failed to run man: $!";
    my ($line1, $lineM) = (scalar(<FH>) || "", "");
    while ( <FH> ) {  # parse manpage into sections
       $line1 = $_ if $line1 =~ /^\s*$/;
       $manpage .= $lineM = $_ unless /^\s*$/;
       if (s/^(\w(\s|\w)+)// || s/^\s*(NAME)/$1/i){
          chomp( my $sec = $1 );  # section title
          $h{$cur_section} .= $cur_content;
          $cur_content = "";
          $cur_section = $sec; # new section name
       }
       $cur_content .= $_ unless /^\s*$/;
    }
    $h{$cur_section} .= $cur_content;

    # examine NAME, HEADer, FOOTer, (and
    # maybe the filename too).
    close(FH) or die "Failed close on pipe to man";
    @h{qw(A_AHEAD A_BFOOT)} = ($line1, $lineM);
    my ($mn, $ms, $md) =
("","","","");
    # NAME mn, DESCRIPTION md, & SECTION ms
    for(sort keys(%h)) { # A_AHEAD & A_BFOOT first
       my ($k, $v) = ($_, $h{$_}); # copy key&val
       if (/^A_(AHEAD|BFOOT)$/) { #get sec or cmd
           # look for the 'section' in ()'s
          if ($v =~ /\(([^)]+)\)\s*$/) {$ms||= $1;}
       } elsif($k =~ s/^\s*(NOSECTION|NAME)\s*//) {
          my $namestr = $v || $k; # 'cmd - a desc'
```

```
            if ($namestr =~ /(\S.*)\s+--?\s*(.*)/) {
               $mn ||= $1 || "";
               $md ||= $2 || "";
            } else { # that regex could fail.
               $md ||= $namestr || $v;
            }
         }
      }
      if (!$ms && $file =~ m!/man/man([^/]*)/!) {
         $ms = $1; # get sec from path if not found
      }
      ($mn = $file) =~ s!(^.*/)|(\.gz$)!! unless $mn;
      my %metas;
      @metas{qw(swishtitle sec desc page)} =
         ($mn, $ms, $md, $manpage);
      return ( \%metas ); # return ref to 5-key hash.
   }
```

The first for loop in Listing 1 is the main loop of the program. It looks at each man page, parses it as needed, converts it to XML and wraps it in the appropriate headers for SWISH-E:

- get_man_file() uses File::Find to traverse the man directories to find man page source files.
- make_xml() and escape() together create XML from the hashref returned by parse_man().
- parse_man() performs the nitty-gritty work of getting the relevant fields from the man page source.

Now that we've explained it, let's use it:

```
% swish-e -c sman-index.conf -S prog
```

When that's done, you can test the index as before, using swish-e's -w option.

As our final example, we discuss a Perl script that uses SWISH::API to use the index we just built to provide an improved version of the UNIX standby apropos. The code is included in Listing 2 (sman). Here's a brief rundown: lines 1-14 set things up and parse command-line options, lines 15-23 issue the query and do cursory error handling and lines 24-39 present the search results using Properties returned through the SWISH::API.

Listing 2. sman is a command-line utility to search man pages.

```
#!/usr/bin/perl -w

use strict;
use Getopt::Long qw(GetOptions);
use SWISH::API;

my ($max,$rankshow,$fileshow,$cnt) = (20,0,0,0);
my $index = "./sman.index";
GetOptions( "max=i"   => \$max,
            "index=s" => \$index,
            "rank"    => \$rankshow,
            "file"    => \$fileshow,
);
my $query = join(" ", @ARGV);
```

```
my $handle = SWISH::API->new($index);
my $results = $handle->Query( $query );
if ( $results->Hits() <= 0 ) {
    warn "No Results for '$query'.\n";
}
if ( my $error = $handle->Error( ) ) {
    warn "Error: ",  $handle->ErrorString(), "\n";
}
while ( ($cnt++ < $max) &&
(my $res = $results->NextResult)) {
    printf "%4d ", $res->Property( "swishrank" )
       if $rankshow;
    my $title = $res->Property( "swishtitle" );
    if (my $cmd = $res->Property( "cmd" )) {
       $title .= " [$cmd]";
    }
    printf "%-25s (%s) %-30s", $title,
       $res->Property( "sec" ),
       $res->Property( "desc" );
    printf " %s", $res->Property( "swishdocpath"
)
       if $fileshow;
    print "\n";
}
```

The Perl client is that simple. Let's use ours to issue searches on our man pages such as:

```
% ./sman -m 1 boot disk
```

We should get back:

```
bootparam (7) Introduction to boot time para...
```

But we now also can do searches like:

```
% ./sman sec=3 perl
```

to limit searches to section 3. The sman program also accepts the command-line option --max=# to specify the maximum number of hits returned, --file to show the source file of the man page and --rank to show each hit's rank for the given query:

```
% ./sman --max=1 --file --rank boot
```

This returns:

```
1000 lilo.conf (5) configuration file for lilo
    /usr/man/man5/lilo.conf.5
```

Notice the rank as the first column and the source file as the last one.

An enhanced version of the sman package will be available at joshr.com/src/sman/.

Conclusion

SWISH-E has two downsides we should mention. First, it's not multibyte safe—it handles only 8-bit ASCII data. Second, records cannot be deleted from a SWISH-E index—to remove records, an index must be re-created. On the plus side, SWISH-E has numerous features we didn't even get to mention. See the SWISH-E web site at www.swish-e.org for more details. We hope you'll agree that SWISH-E is an impressive toolkit and a useful addition to your programming toolbelt.

Josh Rabinowitz is a 13-year veteran of the software industry who cut his teeth at NASA Ames Research Center and at CNET.com and other web companies. He currently is an independent consultant and the publisher of SkateboardDirectory.com, which aims to be your guide to skateboard sites on the Internet.

Archive Index Issue Table of Contents

Advanced search

wxWindows for Cross-Platform Coding

Taran Rampersad

Issue #111, July 2003

The well-tested wxWindows is the toolkit behind the upcoming Chandler groupware.

Multiplatform development has become a mainstream phrase over the last few years, from the halls of Redmond to open-source initiatives. The uses of this phrase all seem to revolve around things like Java, .NET, V and Qt; however, let's not forget wxWindows.

Whether you're porting some MFC code to Linux or simply want to make your application available to the broadest user demographic, wxWindows fits the bill. Strangely, not many people seem to have heard of it, but hopefully, this article assists in making the information more available.

Why choose wxWindows over these other development tools? That's a fair question, and the answer is different for each one.

Compared to the Different Tools

Although Qt is the standard for building KDE applications, wxWindows can be used as well. Qt's Microsoft Windows version is not free for commercial use, but wxWindows is, and Qt requires a special preprocessor for the event system.

Microsoft's .Net and Ximian's Mono, as well as Java, aren't really toolkits as much as they are technologies. These technologies add a layer to the application that can rob an application of performance. This can be debated as a “development time vs. performance” issue and is different depending on the life cycle of the application. Instead of getting into yet another language war, we'll simply say, “your mileage may vary.”

V is simply not as flexible as wxWindows; it doesn't support as many compilers.

In comparing wxWindows to these other multiplatform development tools and languages, it's important to mention that wxWindows stands on its own merits and is not limited to the comparisons.

With wxWindows, you gain the usual open-source freedoms—the freedom to use without charge, the freedom to modify and the freedom from the risk of vendors disappearing (or updating their libraries or tools so that you have to buy a new one). Plus, you have the freedom to run the application on a variety of operating systems, including UNIX/Linux, Windows, Mac OS 9, Mac OS X and OS2. What's more, it gives you a native look and feel on each operating system. The license for wxWindows is flexible and has been approved by the Open Source Initiative.

With about 11 years behind it, wxWindows is a stable, mature development library with real-time support through the main mailing lists found on the web site, wxWindows.org. Many features are available through more than 300 classes and 5,000 functions. Porting an application from the MFC is also straightforward with this library.

Applications That Use wxWindows

Some major companies are using wxWindows, including AMD, the Intel Graphics Lab, the Compaq Alpha Microprocessor Development Group, Netscape and Lockheed Martin. The Open Source Applications Foundation is implementing it (using wxPython) in their new program, Chandler, for managing personal information and collaborating with others.

Helpblocks (www.helpblocks.com) is an application that creates multiplatform Help files and is developed with the wxWindows library running on both Windows and Linux. Other applications include Vulcan 3-D (modeling software for the mining industry, and more), Display Doctor from SciTech, Intuitive MX (a multitrack audio mixer with 3-D representation of the mixing stage, by Intuitive Works) and Ground Control Station from Geneva Aerospace. Your application could be next on this list.

Beginning Linux Development with wxWindows

You'll need GCC and also can use tools such as gdb, ddd, Emacs and an IDE such as Anjuta. To compile on Windows, you may need a cross-compiler. Most importantly, you'll need the wxWindows library itself, which also is available at the wxWindows web site as a download or CD purchase. The latest stable release at the time of this writing is version 2.4.0.

Despite what many of us do, it's a good idea to read the documentation relevant to Linux development that is found in the library package.

Also, it would be wise to subscribe to the wx-user mailing list (found under Support at wxWindows.org), in case you run into difficulties. The most reasonable thing to do is search the archives to see if your question already has been asked.

The library itself comes with 70+ example applications that demonstrate almost everything imaginable you would want to do with it; these in themselves are probably the best documentation because you can tinker with a working application and see how it was done. Once you get the library and become familiar with how to use it with your tools, you're ready to start coding.

A Sample or Two

Creating a simple application class is easy. Listing 1 provides a basic sample of creating a wxApp class.

As you can see, creating a main wxWindows application is rather easy, and those of you familiar with MFC may be rubbing your chin thoughtfully. You'll be glad to know there is no MFC in it, nor will there ever be MFC in it, but the classes are a simple transition from MFC. It has few things MFC doesn't, and it does not support OLE.

Listing 1. Creating a wxApp Class

An empty Event table is present in Listing 1, which allows you to handle events, such as a mouse click or a key pressed or your own custom events. Of course, this code shows a rather boring main application that sits there and stares at you without even blinking.

Working examples abound with the library; every single class appears to have an example demonstrating its usage. One example is the event-handling sample, by Vadim Zeitlin. This example is used because many developers beginning to use the wxWindows library seem to have some problems with events; you'll now be ahead of the game. You need the wxWindows library to try this, and if you do, you get the full source code for it under the Samples directory.

There's even a Wiki linked to from the site, so you can access the latest documentation quickly and even correct it if you find something to add.

Give the Developers Their Due

Aside from the cast of thousands, a core development team should receive special mention and thanks:

- Julian Smart, www.anthemion.co.uk
- Robin Dunn, www.python.org
- Vadim Zeitlin, www.tt-solutions.com
- Stefan Csomor, www.advanced.ch
- Vaclav Slavik, sourceforge.net/users/vaclavslavik
- Robert Roebling, www.roebling.de

These developers frequently are seen helping users of the wxWindows library on a daily, if not hourly, basis on the wx-user mailing list.

wxWindows: the Future

The release of version 2.4.0 of the wxWindows library was followed by requests from the development team for what they would like to see in version 3.0. New ports are always in the works, and the next release should be no different. One of the ports expected is the Windows CE port from Marco Cavallini and Robert Roebling.

Winelib support also is on the way, courtesy of the Winelib team, and proprietary tools for porting from MFC to wxWindows are being worked on by Julian Smart and Stefan Csomor.

With more and more companies wanting to use Linux, they will be looking for ways to port their MFC code to Linux, and one of the simplest is with the wxWindows library. The cost alone makes it worthwhile. Development issues for porting from the MFC are minimalized due to the simplicity of the library itself. Thus, if a company is looking to change from a Windows-based system to a Linux-based system, they can expect an easy porting of any code that they have. A wxNet port is even in progress.

Perhaps you have something to add to the many applications done with wxWindows. An up-to-date list of wxWindows applications is available at wxWindows.org.

email: cnd@knowprose.com

Taran Rampersad is a software developer with 14 years of experience, presently doing consultation, development and writing from Trinidad and Tobago. He's actively involved with local computer societies, process management and assisting in organizing the Caribbean FLOS Conference (floscaribbean.org). His personal web site is KnowProSE.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

An Event Mechanism for Linux

Frederic Rossi

Issue #111, July 2003

Telecom applications have extraordinary requirements for low latency and high complexity. An asynchronous event mechanism can be the basis of meeting them.

This article is the first in a series describing a new event-based mechanism for Linux. This particular article focuses on the motivation, requirements and benefits of such a mechanism for carrier-grade Linux.

The work of supporting a native event-based system in the Linux kernel started as a research project in 2001 at the Open Systems Lab (Ericsson Research, Corporate Unit) in Montréal, Canada. The goal was to provide Linux with an event-driven environment that could deliver better performance compared to existing solutions in the context of telecom applications.

There has been a growing interest in bringing the Linux operating system to a carrier-grade level. For example, the OSDL Carrier-Grade Linux working group (www.osdl.org/projects/cgl) currently is drafting a set of requirements to turn Linux into a solid carrier-grade server for Next Generation Networks.

Operating systems for telecom applications must ensure that they can deliver a high response rate with minimum downtime—less than five minutes per year, 99.999% of uptime—including hardware, operating system and software upgrade. In addition to this goal, a carrier-grade system also must take into account such characteristics as scalability, high availability and performance.

For such systems, thousands of requests must be handled concurrently without impacting the overall system's performance, even under high load. Subscribers can expect some latency time when issuing a request, but they are not willing to accept an unbounded response time. Such transactions are not handled instantaneously for many reasons, and it can take some milliseconds or

seconds to reply. Waiting for an answer reduces applications' abilities to handle other transactions.

Many different solutions have been envisaged to improve Linux's capabilities using different types of software organization, such as multithreaded architectures, by implementing efficient POSIX interfaces or by improving the scalability of existing kernel routines. We think that none of these solutions are adequate for true carrier-grade servers.

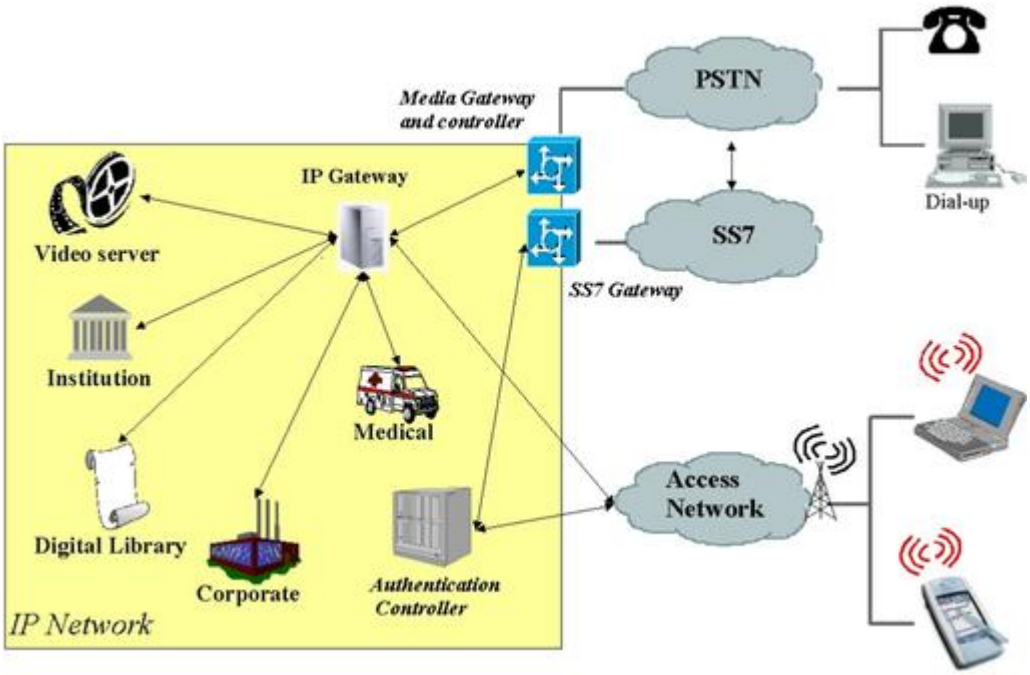


Figure 1. Architecture and Interoperability between the PSTN and IP Networks

In order to understand our point, this article provides an overview of telecom networks. The purpose is to clarify the requirements for a carrier-grade operating system. After this introduction, we explain the benefits of a native asynchronous event mechanism to better support carrier-grade characteristics.

Carrier-Grade Requirements

Telecommunications is concerned with the establishment of telephone calls between two devices and the transport of voice over wire links. Figure 1 shows the traditional Public Switched Telephone Network (PSTN). More specifically, carriers use the term *signaling* to indicate the establishment of a telephone call between two devices. Signaling in the PSTN is done through the Signaling System 7 (SS7) protocol stack. SS7 performs call routing and builds a path to the destination telephone through the circuits. The two phones are connected once the path is established and the voice can be carried over. The SS7 protocol is able to handle call routing, call forwarding and error conditions.

The flexibility, cost performance and continuous growth of the Internet are driving a migration of many telecom services to it. This helps to establish IP technologies as the new standard for all communications services. These two types of networks are based on different technologies and require the utilization of signaling and media gateways for inter-working purposes.

Gateways perform the translation of information between different types of network. For example, an SS7 gateway is used to encapsulate signaling over the IP network through the Stream Control Transmission Protocol (SCTP). A media gateway is used to encode and decode the voice coming from the PSTN network and going to the IP network and vice versa.

The signaling and the media gateways provide connectivity to the IP network for calls coming from the PSTN network. Signaling gateways must perform protocol encapsulation in order to carry the syntax and semantics of SS7 messages over an internet protocol, such as SCTP over IP or UDP over IP. Signaling servers must be able to scale with respect to their system capabilities as the number of concurrent requests increases.

With a media gateway, operators can implement transport of streaming data between the PSTN and the IP networks. The number of connections they can accept depends on their hardware, which can vary in size from one to thousands of interfaces. Media gateways must support a large number of connections in real time.

Authentication, authorization and accounting (AAA) servers maintain databases of user profiles. Typically, one or two AAA servers may contain information about several millions of subscribers belonging to a particular network operator. It is common to observe peaks of thousands of concurrent authentication requests per second. Such variation in the number of connections is difficult to plan for in advance. AAA servers have a critical role of controlling access to the IP network and are not allowed to fail. They require soft real-time capabilities, so they can reply to most requests within a few milliseconds.

Media servers provide specialized resources and services to end users, such as video conferences, video servers, applications and e-mail. An important aspect of these carrier-class systems is scalability. These platforms can accept a linear increase of the number of transactions with respect to the number of processors, interfaces or bandwidth. Telecom operators speak of linear scalability, meaning the cost per transaction or per user should not increase when scaling up a server.

In case of failure or unplanned interruption, carrier-grade platforms can recover automatically or failover to another server through network redundancy procedures. Live software upgrades and hot swap of hardware devices also are part of the 99.999% service availability.

Linux has proven to be stable and consistent over the years, and it is already an attractive option for carriers. In order to become a key element of telecom networks, however, it must be enhanced with components providing these much-needed carrier-grade capabilities.

Matching Performance and Architecture

In the traditional programming model, software components explicitly synchronize with others. This is the common model when a lot of interaction is required. For example, the typical approach is to use select() or poll() to listen to file descriptors. A generic implementation of select scans the entire array of descriptors. This is not scalable because the time it takes to detect activity on a descriptor is proportional to the size of the array. This increases the application latency and leads to a decrease in the overall system performance.

Scanning an array of descriptors or waiting for data consumes processing time. A common idea in the design of efficient algorithms is to handle system events asynchronously. Some examples of mechanisms that provide event notification to user-space applications are the POSIX AIO, epoll or the BSD kqueue (see Resources).

When describing the efficiency of such mechanisms, it is common to compute the average time it takes from the moment an event is detected in the kernel to the moment it is effectively handled by the application. One of the main reasons this is done is that micro-benchmarks for this type of method are not relevant. Such mechanisms can be quite efficient locally but inefficient when combined with other mechanisms not well adapted, such as multithreaded architectures. As an example, many web servers use a pool of threads that is started when the application is launched. A typical architecture is to use one dedicated thread to manage incoming connections and one thread per transaction. Usually this design is efficient for a low number of incoming connections but is inefficient when the load goes higher.

Multithreaded applications are needed when a high level of concurrency is required between objects competing for the CPU. Well-known examples are found in high-performance computing applications where the speed of execution of every thread is important, but the number of threads run is not high.

Threads provide a sequential and synchronous model of development, and they have become the standard way of implementing applications when a high level of concurrency is needed. But flaws in the design of applications or flaws in handling synchronizations easily can create system contentions and affect the overall system performance. J. Ousterhout, in “Why Threads Are a Bad Idea”, established that programming with threads is quite difficult and mainly leads to applications unable to execute properly under high loads.

No competition between threads exists in telecom applications. But concurrency occurs when handling common objects, such as distributed data structures. For these applications, threads are needed to provide concurrent accesses to shared data.

Telecom applications are used to handle thousands of transactions per second and hundreds of simultaneous connections on the same processor. In addition, system events, including database accesses, applications faults, overload notifications, alarms, state change of system components and so on, must be taken into account. Thousands of events can be generated in the same system during the execution of an application, so managing events with threads would be inefficient.

Traditional asynchronous mechanisms try to solve this scalability issue by preventing applications from waiting unnecessarily or, like epoll on Linux, they aim to improve the detection of active descriptors. Unfortunately, these solutions are limited to file descriptors, which represent only a fraction of the events of interest. Also, starting a huge number of threads, as needed for web servers to handle these events, would create a bottleneck and aggravate the situation.

Event-Based Architectures

The development of complex distributed software architectures demands the implementation of a mechanism that is suitable to take advantage of system resources at runtime. A promising solution that is more appropriate to address this issue is the introduction of an event-based mechanism in Linux. Such mechanisms enable a real cooperation between the operating system and the applications. They provide components able to register for events that can be asynchronously notified later, through the execution of handlers.

If we compare signal handlers and event handlers, we find the latter more informative because they bring the data directly to the application. Basically, an asynchronous event mechanism can be used to implement generic user-level handlers triggered by system events or to implement periodic monitoring components, like timers. The first case is particularly interesting if an application doesn't know when an event occurs. When receiving events

asynchronously, the application can take action without recovering all the necessary data because it is supplied in parameters.

Some investigations already have been done regarding fast message-passing mechanisms, which are based on the same principles as asynchronous events. For example, active messages (see Resources) execute asynchronously on the stack of the receiver process. In pop-up threads, a thread of execution is created for every handler, and in single-threaded upcalls, a dedicated thread is created on each processor. AEM is an emerging mechanism that offers a native environment for the development of applications requiring real asynchrony. For example, we used AEM to implement a native asynchronous socket interface for TCP. In AEM, the choice is made at registration time to define a handler that is executed on either the current execution task or a new thread of execution. Some other research projects have proposed similar solutions to improve web server capabilities under high load (see “A Scalable and Explicit Event Delivery Mechanism for UNIX”, Resources).

The main benefit of the event paradigm is the integration of event handling and thread management in the same mechanism. Concretely, it gives full control to resource consumption.

Performance is really a goal for event-based mechanisms. Decoupling event management from the application permits increased locality by taking advantage of different memory allocation schemes or influencing the scheduler decision. For example, soft real-time responsiveness is ensured by enforcing process priorities depending on pending events.

This emerging paradigm provides a simpler and more natural programming style compared to the complexity offered by multithreaded architectures. It proves its efficiency for the development of multilayer software architectures, where each layer provides a service to the upper layer. This type of architecture is quite common for distributed applications.

Figure 2 illustrates a typical distributed application based on an event-driven model. It is composed of many software components, and a process represents one layer of the application. In distributed applications, a lot of local and remote communications are engaged either at the same level or at a different level.

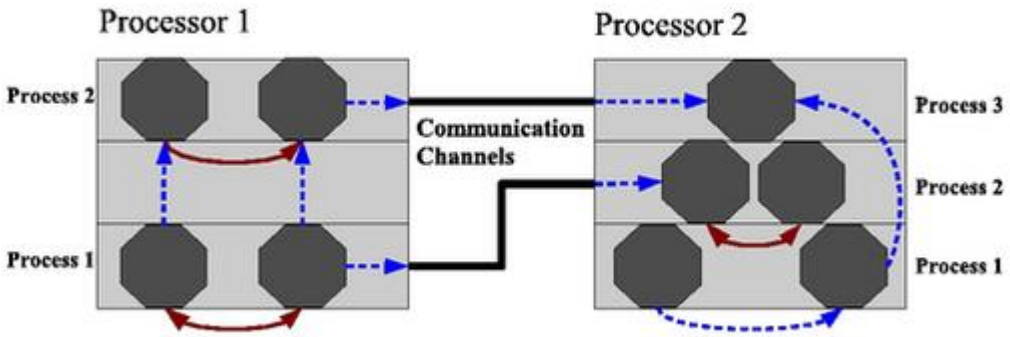


Figure 2. A multilayer distributed application designed with an event-based model running on two processors. Each layer is single-threaded, and the communication between application components is either synchronous (plain lines) or asynchronous (dashed lines).

In many situations such applications have to provide services that must operate worldwide with high performance. It is essential that these applications take advantage of hardware resources and scale linearly with respect to the platform's capabilities.

The design of this software must ensure that no deadlock or race condition is possible between the components. The impact of such design flaws on system integrity can be catastrophic. This situation is difficult to solve when using a multithreaded approach, because it is hard to detect and correct due to the high number of possible configurations. An event-based mechanism reduces the chance of introducing points of failure by controlling the number of threads started asynchronously. It is easier to guarantee atomicity of handler executions, because the mechanism is kept in the kernel.

System resources are limited, and the number of processes that can be started is always limited. At registration time, the alternative is given to choose the type of handler to execute. This permits the production of more robust applications as the load increases. The main advantage for applications is the possibility to mix sequential code and asynchronous code. It then is possible to design applications that exploit capabilities of both strategies.

An event-based framework offers operators dynamic reconfiguration with minimum impact on the system uptime. Hardware hot swap and dynamic software upgrade must be possible without restarting the system. Distributed applications are built from a large number of interacting components, and upgrading such software is a critical operation.

Telecom platforms require 99.999% uptime for all services. The services cannot be stopped during maintenance operations, as this would impact other service platforms and subscriber requests connected to it. Software upgrades must be performed gradually. Event-based mechanisms introduce the potential for such

capability to distributed applications. As we can see in Figure 2, there are no direct dependencies between software layers if communication is performed asynchronously. It then is possible to replace some of the application parts without major disturbance.

Conclusion

An event-based mechanism provides a new programming model that offers software developers unique and powerful support for asynchronous execution of processes. Of course, it radically differs from the sequential programming styles we are used to but offers a design framework better structured for software development. It also simplifies the integration and the interoperability of complex software components.

The strength of such a mechanism is its ability to combine synchronous code and asynchronous code in the same application—or even mix these two types of models within the same code routine. With this hybrid approach, it is possible to take advantage of their respective capabilities depending on the situation. This model is especially favorable for the development of secure software and for the long-term maintenance of mission-critical applications.

In a future article, we will show how AEM has been implemented to provide this support in the Linux kernel and how to use it for software development.

Acknowledgements

The Open Systems Lab for reviewing and approving the publication of this article, Laurent Marchand at Ericsson Research Canada for useful comments and Philippe Meloche, a student at Sherbrooke University.

email: Frederic.Rossi@ericsson.ca

Frederic Rossi is a researcher at the Open Systems Lab at Ericsson Research, Corporate Unit, in Montréal, Canada. He is involved in research activities leading to designing kernel components for the advancement of carrier-class operating systems. He can be reached by e-mail at frederic.rossi@ericsson.ca.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

CPU Affinity

Robert Love

Issue #111, July 2003

Bind specific processes to specific processors with a new system call.

The ability in Linux to bind one or more processes to one or more processors, called CPU affinity, is a long-requested feature. The idea is to say “always run this process on processor one” or “run these processes on all processors but processor zero”. The scheduler then obeys the order, and the process runs only on the allowed processors.

Other operating systems, such as Windows NT, have long provided a system call to set the CPU affinity for a process. Consequently, demand for such a system call in Linux has been high. Finally, the 2.5 kernel introduced a set of system calls for setting and retrieving the CPU affinity of a process.

In this article, I look at the reasons for introducing a CPU affinity interface to Linux. I then cover how to use the interface in your programs. If you are not a programmer or if you have an existing program you are unable to modify, I cover a simple utility for changing the affinity of a given process using its PID. Finally, we look at the actual implementation of the system call.

Soft vs. Hard CPU Affinity

There are two types of CPU affinity. The first, soft affinity, also called natural affinity, is the tendency of a scheduler to try to keep processes on the same CPU as long as possible. It is merely an attempt; if it is ever infeasible, the processes certainly will migrate to another processor. The new O(1) scheduler in 2.5 exhibits excellent natural affinity. On the opposite end, however, is the 2.4 scheduler, which has poor CPU affinity. This behavior results in the ping-pong effect. The scheduler bounces processes between multiple processors each time they are scheduled and rescheduled. Table 1 is an example of poor natural affinity; Table 2 shows what good natural affinity looks like.

Table 1. The Ping-Pong Effect

Table 2. Good Affinitiy

Hard affinity, on the other hand, is what a CPU affinity system call provides. It is a requirement, and processes must adhere to a specified hard affinity. If a processor is bound to CPU zero, for example, then it can run only on CPU zero.

Why One Needs CPU Affinity

Before we cover the new system calls, let's discuss why anyone would need such a feature. The first benefit of CPU affinity is optimizing cache performance. I said the O(1) scheduler tries hard to keep tasks on the same processor, and it does. But in some performance-critical situations—perhaps a large database or a highly threaded Java server—it makes sense to enforce the affinity as a hard requirement. Multiprocessing computers go through a lot of trouble to keep the processor caches valid. Data can be kept in only one processor's cache at a time. Otherwise, the processor's cache may grow out of sync, leading to the question, who has the data that is the most up-to-date copy of the main memory? Consequently, whenever a processor adds a line of data to its local cache, all the other processors in the system also caching it must invalidate that data. This invalidation is costly and unpleasant. But the real problem comes into play when processes bounce between processors: they constantly cause cache invalidations, and the data they want is never in the cache when they need it. Thus, cache miss rates grow very large. CPU affinity protects against this and improves cache performance.

A second benefit of CPU affinity is a corollary to the first. If multiple threads are accessing the same data, it might make sense to bind them all to the same processor. Doing so guarantees that the threads do not contend over data and cause cache misses. This does diminish the performance gained from multithreading on SMP. If the threads are inherently serialized, however, the improved cache hit rate may be worth it.

The third and final benefit is found in real-time or otherwise time-sensitive applications. In this approach, all the system processes are bound to a subset of the processors on the system. The specialized application then is bound to the remaining processors. Commonly, in a dual-processor system, the specialized application is bound to one processor, and all other processes are bound to the other. This ensures that the specialized application receives the full attention of the processor.

Getting the New System Calls

The system calls are new, so they are not available yet in all systems. You need at least kernel 2.5.8-pre3 and glibc 2.3.1; glibc 2.3.0 supports the system calls, but it has a bug. The system calls are not yet in 2.4, but patches are available at www.kernel.org/pub/linux/kernel/people/rml/cpu-affinity.

Many distribution kernels also support the new system calls. In particular, Red Hat 9 is shipping with both kernel and glibc support for the new calls. Real-time solutions, such as MontaVista Linux, also fully support the new interface.

Affinity Masks

On most systems, Linux included, the interface for setting CPU affinity uses a bitmask. A bitmask is a series of *n* bits, where each bit individually corresponds to the status of some other object. For example, CPU affinity (on 32-bit machines) is represented by a 32-bit bitmask. Each bit represents whether the given task is bound to the corresponding processor. Count the bits from right to left, bit 0 to bit 31 and, thus, processor zero to processor 31. For example:

```
11111111111111111111111111111111 = 4,294,967,295
```

is the default CPU affinity mask for all processes. Because all bits are set, the process can run on any processor. Conversely:

```
00000000000000000000000000000001 = 1
```

is much more restrictive. Only bit 0 is set, so the process may run only on processor zero. That is, this affinity mask binds a process to processor zero.

Get it? What do the next two masks equal in decimal? What is the result of using them as the affinity mask of a process?

```
10000000000000000000000000000000
00000000000000000000000000000011
```

The first is equal to 2,147,483,648 and, because bit 31 is set, binds the process to processor number 31. The second is equal to 3, and it binds the process in question to processor zero and processor one.

The Linux CPU affinity interface uses a bitmask like that shown above. Unfortunately, C does not support binary constants, so you always have to use the decimal or hexadecimal equivalent. You may get a compiler warning for very large decimal constants that set bit 31, but they will work.

Using the New System Calls

With the correct kernel and glibc in hand, using the system calls is easy:

```
#define _GNU_SOURCE
#include <sched.h>
long
sched_setaffinity(pid_t pid, unsigned int len,
                  unsigned long *user_mask_ptr);
long
sched_getaffinity(pid_t pid, unsigned int len,
                  unsigned long *user_mask_ptr);
```

The first system call is used to set the affinity of a process, and the second system call retrieves it.

In either system call, the PID argument is the PID of the process whose mask you wish to set or retrieve. If the PID is set to zero, the PID of the current task is used.

The second argument is the length in bytes of the CPU affinity bitmask, currently four bytes (32 bits). This number is included in case the kernel ever changes the size of the CPU affinity mask and allows the system calls to be forward-compatible with any changes; breaking syscalls is bad form, after all. The third argument is a pointer to the bitmask itself.

Let us look at retrieving the CPU affinity of a task:

```
unsigned long mask;
unsigned int len = sizeof(mask);
if (sched_getaffinity(0, len, &mask) < 0) {
    perror("sched_getaffinity");
    return -1;
    }
printf("my affinity mask is: %08lx\n", mask);
```

As a convenience, the returned mask is binary ANDed against the mask of all processors in the system. Thus, processors in your system that are not on-line have corresponding bits that are not set. For example, a uniprocessor system always returns 1 for the above call (bit 0 is set and no others).

Setting the mask is equally easy:

```
unsigned long mask = 7; /* processors 0, 1, and 2 */
unsigned int len = sizeof(mask);
if (sched_setaffinity(0, len, &mask) < 0) {
    perror("sched_setaffinity");
}
```

This example binds the current process to the first three processors in the system.

You then can call sched_getaffinity() to ensure the change took effect. What does sched_getaffinity() return for the above setup if you have only two processors? What if you have only one? The system call fails unless at least one processor in the bitmask exists. Using a mask of zero always fails. Likewise, binding to processor seven if you do not have a processor seven will fail.

It is possible to retrieve the CPU affinity mask of any process on the system. You can set the affinity of only the processes you own, however. Of course, root can set any process' affinity.

I Want a Tool!

If you are not a programmer, or if you cannot modify the source for whatever reason, you still can bind processes. Listing 1 is the source code for a simple command-line utility to set the CPU affinity mask of any process, given its PID. As we discussed above, you must own the process or be root to do this.

Listing 1. bind

Usage is simple; once you learn the decimal equivalent of the CPU mask, you need:

```
usage: bind pid cpu_mask
```

As an example, assume we have a dual computer and want to bind our *Quake* process (with PID 1600) to processor two. We would enter the following:

```
bind 1600 2
```

Getting Really Crafty

In the previous example, we bound *Quake* to one of the two processors in our system. To ensure top-notch frame rates, we need to bind all the other processes on the system to the other processor. You can do this by hand or by writing a crafty script, but neither is efficient. Instead, make use of the fact that CPU affinity is inherited across a fork(). All of a process' children receive the same CPU affinity mask as their parent.

Then, all we need to do is have init bind itself to one processor. All other processes, by nature of init being the root of the process tree and thus the superparent of all processes, are then likewise bound to the one processor.

The cleanest way to do this type of bind is to hack this feature into init itself and pass in the desired CPU affinity mask using the kernel command line. We can accomplish our goal with a simpler solution, though, without having to modify

and recompile init. Instead, we can edit the system startup script. On most systems this is /etc/rc.d/rc.sysinit or /etc/rc.sysinit, the first script run by init. Place the sample bind program in /bin, and add these lines to the start of rc.sysinit:

```
/bin/bind 1 1
/bin/bind $$ 1
```

These lines bind init (whose PID is one) and the current process to processor zero. All future processes will fork from one of these two processes and thus inherit the CPU affinity mask. You then can bind your process (whether it be a real-time nuclear control system or *Quake*) to processor one. All processes will run on processor zero except our special process (and any children), which will run on processor one. This ensures that the entire processor is available for our special process.

Kernel Implementation of CPU Affinity

Long before Linus merged the CPU affinity system calls, the kernel supported and respected a CPU affinity mask. There was no interface by which user space could set the mask.

Each process' mask is stored in its task_struct as an unsigned long, cpus_allowed. The task_struct structure is called the process descriptor. It stores all the information about a process. The CPU affinity interface merely reads and writes cpus_allowed.

Whenever the kernel attempts to migrate a process from one processor to another, it first checks to see if the destination processor's bit is set in cpus_allowed. If the bit is not set, the kernel does not migrate the process. Further, whenever the CPU affinity mask is changed, if the process is no longer on an allowed processor it is migrated to one that is allowed. This ensures the process begins on a legal processor and can migrate only to a legal processor. Of course, if it is bound to only a single processor, it does not migrate anywhere.

Conclusion

The CPU affinity interface introduced in 2.5 and back-ported elsewhere provides a simple yet powerful mechanism for controlling which processes are scheduled onto which processors. Users with more than one processor may find the system calls useful in squeezing another drop of performance out of their systems or for ensuring that processor time is available for even the most demanding real-time task. Of course, users with only one processor need not feel left out. They also can use the system calls, but they aren't going to be too useful.

Resources

email: rml@tech9.net

Robert Love is a kernel hacker involved in various projects, including the preemptive kernel and the scheduler. He is a Mathematics and Computer Science student at the University of Florida and a kernel engineer at MontaVista Software. He enjoys photography.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Zope's CMF

Reuven M. Lerner

Issue #111, July 2003

Make your web site's workflow system operate the way your content providers and users want it to work. Develop with the Python-based content management framework.

Over the last few months, we have looked at content management in general and at the open-source Plone content management system (CMS) in particular. Plone certainly is a capable CMS, one that people can use almost immediately upon installation. The main advantage of Plone is its simplicity—it's easy to install, easy to use and easy to customize.

But customizing Plone, or any CMS, can go only so far. If you dislike Plone's boxy look, you undoubtedly can change it in favor of a different model. But if you want a completely different workflow than the one Plone provides, it probably is a waste of time and effort to try to make that sort of change. Rather, it makes sense to write your own CMS or customize one that is designed to be flexible (if more complex) in this area.

Indeed, most CMS vendors recognize that their products need to be customizable if they are going to be useful. If you buy a J2EE-based CMS—and a large percentage of commercial CMS offerings are based on J2EE—you can expect to be able to write new Java objects that describe your content and the way it is published. At a certain point, the division between customizing a CMS and writing your own CMS on an existing infrastructure becomes somewhat blurry.

Enter Zope's content management framework (CMF), designed to provide enough infrastructure for you to create your own CMS. Because Zope development is fairly quick and easy, and because you can use the existing infrastructure that Zope provides, it should be possible to create a CMS at least as quickly (and far more cheaply) than would be possible using a commercial

CMS implementation. Plone is implemented on top of CMF, rather than on its own, meaning each time CMF is improved, Plone benefits from those changes.

This month we look at CMF, which is becoming the central focus of the Zope application server. Indeed, although the latest stable version of CMF is 1.3.1, an alpha version of CMF 2.0 is now available. And, if Zope's future directions were any mystery before now, it is clear that CMF 2.0 is "a lightly repackaged head checkout of Zope3".

Installing and Configuring CMF

We covered how to install CMF several months ago [see At the Forge, *LJ*, May 2003], so I won't go into detail here. In short, download the latest version from cmf.zope.org and unpack the tarfile into lib/python/Products under your Zope home directory. Then, make symbolic links from the Products directory into the CMF directory for CMFCore, CMFDefault, CMFTopic and CMFCalendar. Restart Zope, and you should see a number of CMF-related products appear in the Add menu within the Zope management screen.

Before we can create any CMF objects, we first need to create a container in which our CMF site can exist. You might notice an obvious parallel here between creating a Plone site and a CMF site. To create a new CMF site, simply choose CMF Site from the Add menu in the web-based Zope management interface. You are asked to name the CMF site, as well as to provide a description.

When you create a CMF site, you also are asked if you want a new user folder within that CMF site or if you want to use an existing user folder. For now, use the existing user folder, meaning that users defined within the top-level Zope site are users within the CMF site. If you prefer to make your CMF site a self-contained unit, without reference (except for the site owner) to the outside world, you may want to create your own user folder.

When you have finished creating your CMF site, you are taken to the home page, which tells you to visit the basic configuration form. Because the CMF originally was known as the Portal Toolkit (PTK), many of the screens refer to portals rather than CMF sites. The information you enter in this form is fairly general in nature, allowing you to set, for example, the e-mail addresses from which generated e-mail appears to come as well as the site's SMTP server.

Things get much more interesting if you follow the directions and go to the CMF management interface, which is really the Zope management interface for the CMF site. In other words, if your CMF site is known as /cmfdemo, you can look at the contents of the site with /cmfdemo/manage. The management screen, as usual in Zope, consists of a small navigation bar on the left. But, as we saw last

month, the left side contains a number of portal tools, allowing us to configure and modify our CMF site.

Clicking on portal catalog reveals a vocabulary, Zope's term for an index, that explains how CMF sites are able to provide full-text search without any effort from the site administrator. Clicking on portal types reveals a list of content object classes. These classes form the core of CMF. We look at the content types in greater detail below and will examine how to create our own content types next month.

Finally, click on portal_workflow, which allows you to enter the title of the workflow object you would like to use for each content type. Workflow describes how content moves from writing to publishing, ensuring that only appropriate people are given the authority to perform certain tasks. Authors may write stories, for example, but be unable to publish them to the site. A good workflow system allows you to customize these rules to reflect your organization's needs.

A Simple CMF Site

Now that we have examined the CMF control panel, it's time to examine our site. Upon entering a bare CMF site, we see a main content area in the middle, with several toolbars and boxes in various places. The topmost menu has main navigation links for moving to the top of the site to member pages to the news page and for searching through the site's contents. Underneath that menu, but still in the upper-right corner, is a list of user-specific menus, beginning with My Preferences. This allows logged-in users to set their own preferences, add links to their personal list of favorites and log out. Users who have not logged in to the system are invited to do so if they already have an account or to join the system as a member if they do not yet have an account. On the left is a navigation menu that lists available folders and allows you to set up certain features, such as syndication and local roles.

If you're used to looking at Plone sites, the default CMF site might look a bit spartan but largely familiar. This is because the default CMF site is designed to be used within a custom CMS; even though it is completely functional, it is not designed to be used in real life. By defining new content types and modifying the display skins, you can have a CMS running in almost no time. And because the display logic is separate from the rest of the system, it is possible to change the look and feel relatively quickly.

Every member of a CMF site can be assigned to one or more roles: Member, Reviewer, Manager or Owner. All of these, except Reviewer, should be familiar to experienced Zope users and administrators. The additional role is necessary

for handling workflow, in which reviewers must approve content before it is published to the Web.

Administrators are shown an extended menu on the left side of the screen, allowing them to look at a content view of the current folder, which lets them view or modify existing content or create new content within the folder. When you create a new object, you not only assign it an ID (what traditional web systems call a filename) and content, but also metadata that describes the content. Although you cannot change an object's type after it has been instantiated, you can change all of its parameters by returning to the content view and opening the content in question.

Each piece of content must be published before it can be visible to others. By default, new content has a status of private, but it can be published by clicking the publish link on the left menubar. Using the same interface, the site administrator can revoke an article from the site's published list. This is a great improvement over traditional web sites, where we remove links or delete files.

In addition, most content types can have discussions optionally attached to them. This is similar to the Comment on This Posting feature so popular on weblogs, allowing site visitors to add their comments to what the official site administrators have written. As you add a piece of content, you can decide if you want to accept the default site definition for discussions or if you would like to override the site-wide setting specifically for one piece.

Content Types

Exactly what are these content types that we can instantiate? Most of them are defined in the CMFDefault product, in individual .py files within lib/python/Products/CMFDefault. This product defines both the configuration tools we saw earlier in the Zope management interface and also the basic content types, such as NewsItem, Portal, Image and Link, that we can instantiate from within the CMF.

If you're like me, you are surprised and impressed by the small size of most of the default content types defined in the CMF. They range from 100 lines at the low end for NewsItem to under 350 lines for Portal. This not only means it is easy to debug and change these content types if issues come up but that adding a new content type is relatively easy.

Indeed, a number of new content types for CMF have been developed, and it seems to be a growing field. For example, if you visit the CMF Collective Project at collective.sf.net, you can see a number of CMF-related products that have been released in recent months. For example, fledgling CMF products are available for ecommerce, photo albums and weblog creation. As CMF becomes

increasingly popular, you can expect to see the CMF Collective similarly grow in popularity.

Conclusion

Because Zope Corp. has said repeatedly that CMF is the future of Zope, and because installing a CMS can be so outrageously expensive, it is clear that Zope Corp. seriously is trying to underbid and outperform its proprietary counterparts. However, because Zope and CMF are open source, we can use them in our own projects both to learn about content management and to edit and publish different items. Next month, we dive in a bit more deeply, looking at how to write your own sample CMF content type.

Reuven M. Lerner (reuven@lerner.co.il) is a consultant specializing in open-source web/database technologies. He and his wife, Shira, recently celebrated the birth of their second daughter, Shikma Bruria. Reuven's book Core Perl was published by Prentice Hall in early 2002, and a second book about open-source web technologies will be published by Apress in 2003.

Archive Index Issue Table of Contents

Advanced search

Exploring Strange New Languages

Marcel Gagné

Issue #111, July 2003

Look up the right word, learn how to pronounce it and translate what you want to say into Hungarian.

It is all about understanding each other, François. Whenever we open up a dialogue with someone from another country, we are making an attempt to establish a common ground for communication, usually choosing one language both can understand or, if that fails, translating back and forth.

Quoi? Of course, you are right, *mon ami*. Sometimes it is difficult to make yourself understood even by those who share your language. Thinking you know what a word means is no guarantee the person you are talking to interprets that word in the same way. That is one of the reasons we have dictionaries—that and *Scrabble*.

Ah, *mes amis*! It is good to see you all. Welcome to *Chez Marcel*, home of fine Linux fare and great wines from the world over. Please sit and be comfortable. François and I were discussing the challenges of being understood and of properly getting your meaning across. François, as you already know all this, quickly go down to the wine cellar and bring back the 1999 Napa Valley Cabernet Sauvignon we were tasting earlier—or rather submitting to quality control.

Words are important and the *right* words even more so, as every writer can tell you. This especially is true when you are trying to communicate with someone who doesn't share your language. Using your Linux system, you can take some joy in knowing that you are helping to improve understanding between yourself and others.

The meaning of words, true or otherwise, may be no more than a click away. If you are running KDE 3.0 or higher, try this trick. Let's say you want to find the definition of “cooking”. Open up Konqueror, then type **dict: cooking** in the

Location field. Press Enter, and Konqueror does a search for you in the *Merriam-Webster Online Dictionary*. To do a thesaurus lookup, type **ths: cooking** instead.

KDE has a nice, integrated dictionary application called **Kdict**, part of the kdenetwork package. You'll most likely find Kdict in your Utilities menu under the KDE application launcher (the big K). You also can launch it from the shell with the program name **kdict** (Figure 1). Enter and Kdict connects to various on-line dictionaries to pull up the appropriate definition. Those resources include the *Merriam-Webster Dictionary*, *Wordnet*, *The Jargon File*, *The Devil's Dictionary* and others.

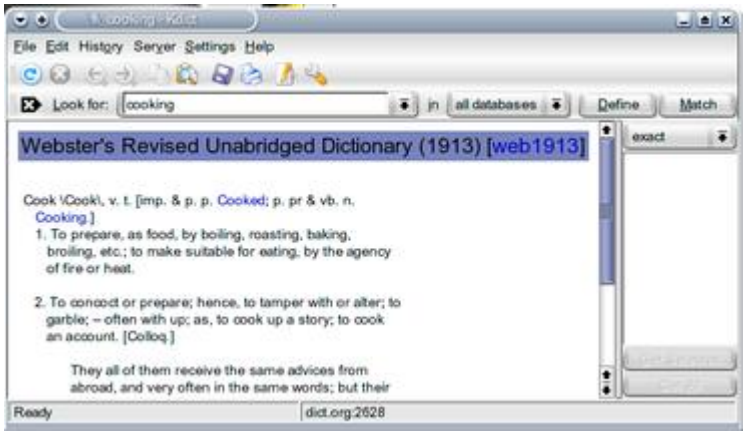


Figure 1. Kdict provides for easy on-line dictionary lookups.

For rapid-fire access to Kdict, you can pop a handy little applet into your Kicker panel. Here's how: right-click on the big K, select Panel menu®Add®Applet®Dictionary. Now, you should see a new program applet labeled Dictionary with three small buttons to the top right on your Kicker panel. On first start, only the C button is visible (define selected text), and the other two are grayed out.

Figure 2. Rapid-Fire Access to Kdict

Enter text into the small window, either a word or a phrase, press Enter, and Kdict appears with that definition as collected from the various sources. You also can select (double-click) a word on a web page or document you are viewing and click that C button in the applet. Kdict automatically launches and provides the definition for the selected word.

If you aren't running KDE or if you prefer a simpler approach, may I interest you in a lightweight, text-only client that does a similar thing? It's Vishal Verma's **edict**; I wonder if he looked that word up in the dictionary. You can get edict

from edictionary.sourceforge.net. The edict program is nothing more than a Perl script, but it does the job quite nicely. In terms of installation, there really isn't much to do after extracting the tarred and gzipped bundle. You can run the script from the directory in which it was extracted, but you'll more than likely want to run a **make install** to save the script to /usr/bin.

To run the program, type **edict** followed by the word you want to look up. For a synonym lookup, type **ethes** followed by a word. If the word you are looking for isn't found, alternatives are offered.

The ethes program is simply a symbolic link to edict. Consequently, a thesaurus lookup is essentially the same process but with different results:

```
[marcel@mysystem edict]$ ethes program
edict - Your personal command line dictionary.
Verison 1.0.
Looking up "program" in Merriam-Webster Online
Thesaurus...
Entry Word: program
Function: noun
Text: 1 a formulated plan listing things to be done
or to take place especially in chronological order
<the program of a concert>
Synonyms: agenda, calendar, card, docket,
programma, schedule, sked, timetable
Related Words: bill; slate; plan
Idioms order of the day 2
Synonyms: COURSE 3, line, policy, polity, procedure
```

Speaking of dictionaries, what is there to say about speaking dictionaries? More to the point, what is there to say about Jeffrey Clement's **MWSpeaker**, which he describes as the “worst speech synthesis software ever”? Those are his words, *mes amis*, not mine. The idea is simple, if not a bit silly. You type in a word or a phrase and MWSpeaker reads it back in human speech. The speech in question comes from the *Merriam Webster Online Dictionary*. In short, it finds each word's corresponding wav file, downloads it and plays it in sequence.

Given that MWSpeaker is a Python script, it requires no compiling per se. Simply download the program from www.jclement.ca/Projects/mwspeaker, then unpack the tarred and gzipped bundle. Before you actually can use the program, you need a few additional packages, most notably wxPython, pygame and PythonCard. To run the program, **cd** to mwspeaker-1.0 (where you unpacked MWSpeaker), and type the following:

```
mkdir data
python mwspeaker.pyw
```

The data directory is where the wav files are stored. The GUI is simple. Type a word or phrase, click Say it and wait. I say “wait”, because MWSpeaker downloads each word's wav file in turn before playing your selection. The

results can be a lot of fun because the voices saying the words aren't consistent. You wind up with a strange mix of male and female voices.

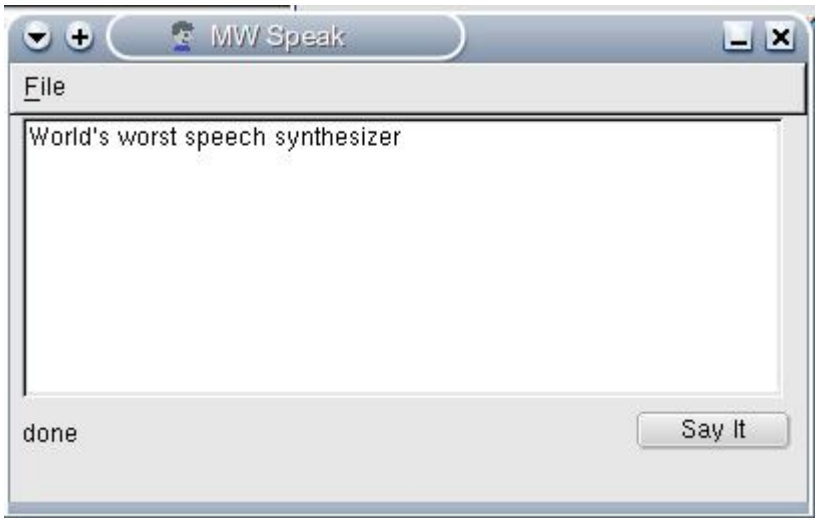


Figure 3. The Self-Proclaimed World's Worst Speech Synthesizer

All this is wonderful for the English language, but Linux and open-source developers come from every part of the world, after all, as do Linux users. It is true that an English language dictionary is useful to those who don't count English as their first language, but sometimes you must translate.

François, *remplisser les verres de nos invités, s'il vous plâit*.

To translate French (or Italian, or Spanish, or German and so on) into English, you might find yourself looking for a Babel Fish. What is a Babel Fish, you ask? According to Douglas Adams, the creator of the *Hitchhiker's Guide to the Galaxy*, it is a small yellow fish that, when put in your ear, simultaneously translates any language you hear into the one you normally speak. But as Douglas Adams wrote, “Meanwhile, the poor Babel Fish, by effectively removing all barriers to communication between different races and cultures, has caused more and bloodier wars than anything else in the history of creation.”

KDE's combination web browser, file manager and Swiss Army knife, Konqueror, has hooks built in to AltaVista's Babel Fish. Surf on over to a foreign language web site, and you easily can translate the information you find there. For my example, I randomly picked a German language newspaper, *Die Welt*, which I learned means “The World”.

When your page has loaded, click Tools on Konqueror's menubar, select Translate Web Page, then select from one of the language selections in the drop-down list. In my example above, I chose “German to English”, *et voilà*! You now can read the information in a language that makes more sense to you.

For your own personal and local translation dictionary, you may want to consider taking a look at Ricardo Villalba's **wordtrans** at wordtrans.sourceforge.net (Figure 4). Compiling wordtrans can be a little tricky, but it isn't a great problem. A visit to TuxFinder (www.tuxfinder.org) turns up quite a number of precompiled packages. I downloaded both the base wordtrans package along with the wordtrans-kde packages in RPM format and installed them. If you are downloading packages, you do need both. You may also find a wordtrans-qt and a wordtrans-web package.

Figure 4. wordtrans, Your Personal Translation Dictionary

By default, wordtrans comes with English, French, Italian, Portuguese and Spanish translation dictionaries, but more can be added. You'll find links to other language files on the wordtrans web site and in the software itself. When you start up Kwordtrans, it may appear as though nothing happened, but look at your system tray in the Kicker panel and you'll see a little gray book icon. Click here and the Kwordtrans interface appears. To select your language of choice, click Dictionaries in the menubar and choose from the list. Choose the direction of your translation (for example, English to Spanish or Spanish to English), type in a word and press Enter.

As I mentioned, you can add additional dictionaries by clicking View on the menubar and selecting Introduction for some links. This not only extends wordtrans' capabilities, but some of the dictionaries available for download are more extensive than the default ones. To add a downloaded language file, click Dictionaries on the menubar, select New and follow the instructions. I downloaded and installed several from www.linuks.mine.nu/dictionary with excellent results.

I'm afraid I do not speak Hungarian, *mes amis*, but apparently I would have to say *az idõ lejárt*, which my desktop translator tells me means “time's up”. It is indeed closing time, but there is time enough for another glass of wine before you go. Raise your glass and my faithful waiter, François, happily will take care of you. As you can see, with a little exploration in our Linux kitchens, we may one day be able to communicate effortlessly with the world. Until next time, *mes amis*, let us all drink to one another's health. *A votre santé*! *Bon appétit*!

Resources

Marcel Gagné lives in Mississauga, Ontario. He is the author of Linux System Administration: A User's Guide (ISBN 0-201-71934-7), published by Addison-Wesley, and is currently at work on his next book. He can be reached via e-mail at mggagne@salmar.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

LDAP for Security, Part I

Mick Bauer

Issue #111, July 2003

OpenLDAP offers the convenience of a common directory across all applications. And if you set it up right, it will make your network more secure, not less.

Suppose you have an Internet Mail Access Protocol (IMAP) server and a bunch of users, but you don't want to give each user a shell account on the server. You'd rather use some sort of central user-authentication service that can be used for other tasks as well. While you're at it, you also need an on-line address book for your organization's e-mail and groupware applications. And suppose, in addition to all that, you also need to provide your users with encryption tools that use X.509 certificates and then manage digital certificates for your entire organization.

Would you believe that one service can address all four scenarios? The Lightweight Directory Access Protocol (LDAP) does all of this and more. And wouldn't you know it, the Open Source community is blessed with a free, stable and fully functional LDAP server and client package that is already part of most Linux distributions: OpenLDAP.

The only catch is LDAP is a complicated beast. To make sense of it, you're going to have to add still more acronyms and some heavy-duty abstractions to your bag of UNIX tricks. But armed with the next few months' Paranoid Penguin columns and a little determination, you'll have the mighty LDAP burro pulling several large plows simultaneously, making your network both more secure and easier to use. In my experience, “more secure” and “simpler for end users” rarely go hand in hand, so I'm excited finally to be covering OpenLDAP in this column.

LDAP Basics

In a nutshell, LDAP provides directory services, a centralized database of essential information about the people, groups and other entities that comprise an organization. As every organization's structure and its precise definition of essential information may be different, a directory service must be highly flexible and customizable. It's therefore an inherently complex undertaking.

The X.500 protocol for directory services is a case in point. It was designed to provide large-scale directory services for large and complex organizations. Accordingly, X.500 is itself a large and complex protocol, so much so that a lightweight version of it was created: the Lightweight Directory Access Protocol. LDAP, described in RFC 1777, is essentially a subset of the X.500 protocol, and it's been implemented far more widely than X.500 itself has been.

X.500 and LDAP are open protocols, like TCP/IP; neither is a standalone product. A protocol has to be implemented in some sort of software, such as a kernel module, a server dæmon or a client program. Also like TCP/IP, not all implementations of LDAP are alike or even completely interoperable (without modification). The particular LDAP implementation we cover here is OpenLDAP, but you should be aware that other software products provide alternative implementations. These include Netscape Directory Server, Sun ONE Directory Server and even, in a limited way, Microsoft Active Directory in Windows 2000 Server.

Luckily, LDAP is designed to be extensible. Creating an LDAP database on one platform that is compatible with other LDAP implementations is usually a simple matter of adjusting the database's record formats, or schema, which we'll discuss next month. Therefore, it's no problem to run an OpenLDAP server on a Linux system that can provide address book functionality to users running, say, Netscape Communicator on Macs.

Getting and Installing OpenLDAP

Being such a useful and important tool, OpenLDAP is included in most major Linux distributions. Generally, it's split across multiple packages: server dæmons in one package, client programs in another, development libraries in still another. This article is about building an LDAP server, so naturally you'll want to install your distribution's OpenLDAP server package, plus OpenLDAP runtime libraries if they aren't included in the server package.

You might be tempted to forego installing the OpenLDAP client commands on your server if no local user accounts will be on it and you expect all LDAP transactions to occur over the network. However, these client commands are

useful for testing and troubleshooting, so I strongly recommend you install them.

The specific packages comprising OpenLDAP in Red Hat are openldap (OpenLDAP libraries, configuration files and documentation); openldap-clients (OpenLDAP client software/commands); openldap-servers (OpenLDAP server programs); and openldap-devel (headers and libraries for developers). Although these packages have a number of fairly mundane dependencies, including glibc, two are required packages you may not have installed already: cyrus-sasl and cyrus-sasl-md5, which help broker authentication transactions with OpenLDAP.

In SuSE, OpenLDAP is provided in the following RPMs: openldap2-client (in section n1 of SuSE versions 7.3 and 8.0); openldap2 (includes both the OpenLDAP libraries and server dæmons and is found in section n2); and openldap2-devel (found in section n2 for SuSE 7.3 and n4 for SuSE 8.0). As with Red Hat, be sure to install the package cyrus-sasl, located in SuSE's sec1 directory.

In both the 7.3 and 8.0 distributions, SuSE provides packages for OpenLDAP versions 1.2 and 2.0. Be sure to install the newer 2.0 packages listed in the previous paragraph, unless you have a specific reason to run OpenLDAP 1.2. This guideline does not apply to Red Hat or Debian, both of which are standardized on OpenLDAP 2.0 in their current distributions.

For Debian 3.0 (Woody), the equivalent deb packages are: libldap2 (OpenLDAP libraries, in Debian's libs directory); slapd (the OpenLDAP server package, found in the net directory); and ldap-utils (OpenLDAP client commands, also found in the net directory). You'll also need libsasl7 from the Debian libs directory.

If your distribution of choice doesn't have binary packages for OpenLDAP, or if a specific feature of the latest version of OpenLDAP is lacking in your distribution's OpenLDAP packages or if you need to customize OpenLDAP at the binary level, you always can compile it yourself from source you've downloaded from the official OpenLDAP web site at www.openldap.org.

Configuring and Starting slapd

The main server dæmon in OpenLDAP is called slapd, and configuring this program is the first step in getting OpenLDAP working once it's installed. Its configuration is determined primarily by the file /etc/openldap/slapd.conf. The “OpenLDAP 2.0 Administrator's Guide” at www.openldap.org/doc/admin20/guide.html has an excellent quick-start procedure for getting slapd up and running: it's in Section 2, starting at Step 8. That document also explains

directory services and LDAP concepts in more depth than this article does, using tree/hierarchy diagrams.

Let's walk through this procedure to make sure you get off to a good start. The first thing to do is edit slapd.conf, an example of which is shown in Listing 1. As you can see, slapd.conf is a typical Linux configuration file: each line consists of a parameter name followed by a value.

Listing 1. Customized Part of /etc/openldap/slapd.conf

The first parameter shown in Listing 1, database, specifies what type of database back end to use. Usually, the best choice here is ldbm, which uses the fast dbm database format, but shell (for custom shell-script back ends) and passwd (to use /etc/passwd as the back end) also are valid choices. There may be multiple database definitions, each with its own set of applicable parameters; all the lines in Listing 1 comprise a single database definition.

The next parameter in Listing 1 is suffix, which determines what queries match this database definition. Here, the specified suffix is wiremonkeys.org, expressed in LDAP-speak as a series of domain component (dc) statements, which are parsed from left to right. In other words, if an LDAP client queries our example server for information about the distinguished name (dn) cn=bubba,dc=wiremonkeys,dc=org, our server matches that query against this database definition, as the dn ends with dc=wiremonkeys,dc=org. See the Sidebar “A Crash Course in X.500 Naming” for more information about distinguished names.

Sidebar: A Crash Course in X.500 Naming

The next two entries in Listing 1 have to do with LDAP database administration; rootdn and rootpw specify the user name and password, respectively, that must be supplied by remote or local commands that perform administrative actions on the LDAP database. Interestingly, this entry is used *only* for this purpose. It doesn't show up in regular LDAP database queries.

This addresses the paradox of how to authenticate the actions required to populate the authentication (LDAP) database. Later, after you've populated your LDAP database with real entity records, designate one of them as the administrative account, using slapd.conf access control lists (acls), and delete the rootdn and rootpw entries. I'll cover that step in a future column; for now, rootdn and rootpw suffice.

It's a very, very bad idea to store the value of rootpw as clear text. Instead, you should use the **slappasswd** command to generate a password hash, shown in Listing 2.

Listing 2. The slappasswd Command

As you can see, slappasswd prompts you for a password and prints that password hashed with the algorithm you specify with the -h option. Be sure to enclose this value in curly brackets—see the slappasswd(8C) man page for a list of valid choices. You can copy and paste slappasswd's output directly into slapd.conf, which is precisely what I did to create the rootpw value in Listing 1.

Getting back to Listing 1, the next parameter in this directory definition is directory. Obviously enough, this specifies in which directory on the local filesystem your LDAP directory should be created. Because /var is the customary place for growing files such as logs and databases, Listing 1 shows a value of /var/lib/ldap. This directory must already exist, and make sure it's owned by OpenLDAP's user and group, usually ldap and ldap. Its permissions should be set to 0700 (**-rwx------**).

Technically, that's enough to get started: you can try starting slapd with your ldap startup script, most likely /etc/init.d/ldap, though this may vary among distributions. I encourage you to start adding practice entries to your LDAP database using the **ldapadd** command—the quick-start procedure I mentioned earlier shows how.

Before you begin managing and querying your LDAP database over the network, however, you'll want to configure and enable TLS encryption. This is important, as the simple authentication method used by OpenLDAP sends authentication credentials over the network unencrypted. But I'm out of space for now, so we'll cover that next month. If you can't wait until then, Vincent Danen explains how in his on-line article “Using OpenLDAP for Authentication”, at www.mandrakesecure.net/en/docs/ldap-auth.php, though it is somewhat Mandrake-centric. I'll also discuss some considerations in determining the structure of your LDAP database and show how to build one. Until then, good luck!

email: mick@visi.com

Mick Bauer, CISSP, is *Linux Journal*'s security editor and an IS security consultant for Upstream Solutions LLC in Minneapolis, Minnesota. Mick spends his copious free time chasing little kids (strictly his own) and playing music, sometimes simultaneously. Mick is author of *Building Secure Servers With Linux* (O'Reilly & Associates, 2002).

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

How Linux Makes Companies Smarter

Doc Searls

Issue #111, July 2003

Vendor tales about “solutions” are fine, but how is Linux changing the way regular companies do and manage information technology?

Two perspectives exist regarding Linux's success in the enterprise: one from the inside and one from the outside. Insiders credit the organization's intelligence, resourcefulness and expertise. Outsiders credit vendors with the same qualities. Credit is due to both, of course, and to some degree each side gives some credit to the other. But the press generally takes the outside view, giving the lion's share of credit to vendors. As a result, little credit goes to the qualities that really account for successful use of Linux in the enterprise.

No two inside stories are the same. Every vendor-side story basically is about how products solve problems—that's why vendors love to call their products “solutions”. But Linux isn't about “solutions”. That's a vendor word. Linux is about resourcefulness and intelligence. Those are customer words. Linux succeeds in the enterprise by helping smart companies make themselves smarter. With Linux and open source, the primary supply and demand sides of the marketplace both reside inside the company.

This coexistence is a hard development for many of us to understand. It's hard for vendors because they're used to a marketplace where the supply side is in control and wide profit margins are the ideal, even if they are no longer the norm. It's hard for the press to understand, because so many of the news stories they write are vendor-vs.-vendor sportscasts, in which the marketplace is a playing field and customers are prizes. It's hard for corporate leadership to understand, because they're so used to playing the prize role in vendor playoffs.

The people who don't find it hard to understand are the people who use Linux to make their companies more efficient, more reliable and more effective. Take the case of LSI Logic, the semiconductor maker. Roland Smith is the director of

Global Operations there, running one of four groups that report to the company's CIO. The company's big switch to Linux happened a while back when Smith's team recommended replacing HP-UX with Linux, increasing performance five- or sixfold and cutting costs by two-thirds. Now, Smith says, “Whenever somebody comes in and wants to talk to the CIO about a new application, or a project or anything new, the CIO's question is, 'Does it run on Linux?”'

This environment has some interesting effects on the flow of help. Smith provides an excellent case of help flowing from customer to vendor:

> In our SAP environment, we recently decided we wanted to begin inserting Linux application servers. So we went out and talked to both Dell and HP. We told them what we wanted from each of them was an eval server—one they think is best suited to run Linux. We wanted four processors and a gig of memory.
>
> A couple of interesting things happened. One was HP fumbled around and took two weeks to come back and say, “What do you really need?” The other was that Dell took three days and shipped us a server. They said, “Here. You've got it for three months. Go tell us what you can do with it.” That was kind of wonderful.
>
> So, then we went to SAP and said, “What do you recommend?” And SAP said, “Well, we think SuSE 8 runs okay. And we think that Red Hat 7.1 runs okay. But we don't really know.” Then we went and talked to Red Hat and they said, “We think it'll run nicely on our Advanced Server 2.1, but we don't really know. So how about you go try it out and tell us?”
>
> So, we built it. We bought the Red Hat application server, built it, loaded SAP on it and put it in our test environment. And it's running incredibly well. We're very happy with it. But it's interesting to me that, of all the vendors out there, only Dell said, “We'll happily give you hardware to try it out on, but we don't know anything more than that.”

Where LSI is concerned, most of the intelligence and leadership is located on the customer side.

Another smart customer is Orbitz, the on-line travel company owned by five major airlines. Leon Chism, chief internet architect for Orbitz, says this about how the company applies its own intelligence internally:

> Our developers are very good at leveraging open-source tools to improve our design, build and software management process. On their own they have built a number of tools that parse logs, store them in a

> PostgreSQL database and do *ex post facto* analysis to find errors and opportunities for improvement. The first I heard of one tool was after it had been used to do analysis and improve some sections of the code successfully several times. We also have a number of developers and engineers using tools like Perl, Python and Jython (we employ an official Jython developer) for their own ad hoc tools to help them monitor site behavior. All of this is in addition to the official production management infrastructure. Generally, after a tool has been developed, tested and proven useful, we migrate it into the rest of the management infrastructure our Network Operations Center uses.

He also sees the use of Linux and open source as competitive advantages for Orbitz:

> Orbitz is a company that leverages open-source projects in its production environment and isn't afraid to be public about it. We currently use over 750 Linux machines in our production environment. Our web servers are Apache running on Linux, our application servers are proprietary servlet engines running on Linux and the back-end "booking engine" is comprised entirely of Java services running on Linux. Lastly, the software that does the low fare searching that is one of the key differentiators between Orbitz and our rivals also is running on Linux.

From the examples of LSI and Orbitz, we have a pretty good idea about how Linux and open-source smarts are applied, both within the enterprise and between the enterprise and its outside suppliers. Now what about the outside view? An excellent example is provided by William M. Bulkeley, in "Out of the Shadows: Open-source software is not only becoming acceptable—It's also becoming a big business"—a feature that ran in the March 31, 2003, *Wall Street Journal*:

> Companies have found ways to make money by providing services to open-source-software users or by packaging this free software with products they sell. With the profit motive driving its promotion, free software is cropping up all over corporate and government computers.

When I first saw that piece, I thought, "Cool!" But then I began to ask some questions:

- Why is it most important that "companies have found ways to make money"? Is that more important than the actual work that gets done?
- Why does he say "companies" when he means "vendors"? Aren't customers companies too?

And let's face it: Linux users are tough customers. Here's Leon Chism again:

> On the relationship front, there is a vendor that we have and are still considering replacing with an open-source solution. For the most part, the application works, but when we find bugs in it the resolution process leaves a lot to be desired. Escalation procedures, arguments about root cause, long discussions about "supported configurations" all lead to muddling the process rather than focusing on the solution.
>
> When we have issues or questions with Linux or Apache, these things aren't at issue. We simply use the customer service application to end all customer service applications—www.google.com—and start researching. Or we start perusing source code. Orbitz is a customer that typically pushes the limits of the products we use. Some vendors are up for the challenge and are willing to make efforts to support that need. Some aren't. And we don't know who is who until, as they say, push comes to shove. Using open-source products removes that issue. We have an engineering and software development staff that is fully capable of paddling its own canoe, and the support we get from authors, list servs and Google is more than enough in most cases.

Yet the tough vendor story gets played while the tough customer story does not.

The Intelligence Imperative

Dell may be the top-selling PC brand, but the real best-sellers are no-name "white" boxes that are bought by the ton or built out of industry-standard parts on an as-needed basis. Customers love them, because they're cheap to buy and easy to replace. But we don't hear much about them. One reason we don't is their builders don't spend much money on advertising and PR. But a bigger reason is the computer industry has a long-standing prejudice against the word "commodity" and has a huge fear of "commodification".

Commodification has long been a fact of life in the hardware business, but software still is widely considered a big margin category. Microsoft has a bad quarter when gross profit margins slip below 80%, and Oracle's worst quarter in the last five (at the time of this writing) was still over 75%.

But the software industry will have to face the fact that customers love commodities. One good example is the University Corporation for Atmospheric Research (UCAR, www.ucar.edu/ucar/index.html) in Boulder, Colorado. In spite of its name, UCAR is a business with substantial customers that include the

national air traffic control system. Greg Thompson, a scientist with UCAR, explains the decision-making process there:

> Everybody here has a shelf full of O'Reilly books. We like to figure things out for ourselves. For example, I'm a scientist, not a programmer. But a while back I saw a need to get past manipulating data with Perl and text files. So I started reading up on databases and learning MySQL. I wouldn't have bothered if MySQL hadn't been free. We don't have five-digit figures around here to run Oracle. We don't need hard-core database stuff. We don't need transaction support. But we do need results that are Web-accessible, so MySQL made sense. Building internal expertise around something like MySQL is straightforward and easy. I do a little research on my own, see if something I need is already installed—and if it's not, I send a help-desk e-mail to a sysadmin. Then, the next day I download a Debian package, run **apt-get** and I'm in business. Without spending a dime.

It's amazing how small a software approval bureaucracy can get. When friction is low and costs are zero, free enterprise lives up to its name. “Our IT budget is zero”, says Elliot Noss, president and CEO of Tucows (tucows.com) in Toronto. Tucows' business is hosting one of the largest software download sites in the world, plus running one of the three top-level domain name registries (OpenSRS, resellers.tucows.com/opensrs), which manages around 3.5 million domain names, all on Linux, Apache, MySQL and PostgreSQL. “I can't even imagine what it would cost to work with a big Sun system or to run a big Oracle database”, Noss says.

Although network and technology companies are more willing to talk about using Linux and open source, other types of companies are quietly changing the way they handle software, too. Take this e-mail, for example, whose author requested anonymity:

> I work for a Fortune 50 company whose IT has a new open-source policy, as well as a fully approved open-source toolkit (with an internal “brand name”) available for employee download from an internal web site. Already up to version 3.0, it comes in both UNIX and Windows flavors. It consists of GNU tools and others on the UNIX side and the Cygwin toolkit on the Windows side. Linux (Red Hat for now) is one of the “tools”.
>
> The guy who heads up the toolkit project is a take-no-prisoners, free-software advocate, too. The stated purpose of the policy is to permit the company to save money by eliminating license fees and by streamlining the software acquisition process, but the clear subtext is to make it possible for the company to protect its

> intellectual property from third-party (that is, Microsoft) manipulation.
>
> Part of the new policy is something of "don't ask, don't tell". IT agrees not to complain if anyone decides to install these tools on whatever computer they use. All any nonmanagement employee now needs to do is get approval from his or her local manager to download and install the tools. Managers do not need anyone's permission to install the tools for themselves. Per the policy, IT does not get involved in any way. So spread of the tools—of Linux and the rest of the free software suite—happens naturally and organically.
>
> Before the policy was implemented, even GNU Emacs required an official IT review/approval process. Even if your boss wanted it, IT could fight it if they wanted to. But with this new policy, that's now history.

Policy inevitably adapts to new facts of development life inside an organization. Today the trends among those facts favor Linux, big time. In a recent survey of Linux developers (who also work with other platforms), Evans Data Corp. observed a rapid shift away from proprietary UNIXes and Windows, in the direction of Linux:

> Linux is the primary choice of host platform at 40%. Windows 2000 makes a strong showing at 29%, with Windows XP right behind at 12%. The landscape is about to change, however....Next year respondents plan to increase their use of Linux as the primary development platform by 15%, from 40% to 55%.

When my coauthors and I wrote *The Cluetrain Manifesto* (www.cluetrain.com) in 1999, we said, "markets are getting smarter—and getting smarter faster than most companies." Four years later, we see something similar happening inside companies with respect to their technology vendors. The result is an ecosystem that subordinates vendors to work. This is good for everybody, including vendors—it means the industry is finally growing up.

Vendors: Giving Customers What They Want

The new imperative for vendors is to give customers what they want and not bother trying to lock them into no-choice relationships. They've had enough of that, thank you. Mårten Mickos, CEO of MySQL (mysql.com), believes vendors like his can help raise IT consciousness. He says, MySQL successfully sells free software (MySQL is GPLed code) because "there is more value to code you can see than to code you can't see", adding, "We are part of a huge community of customers and other developers who are all passionate about improving the code base. Every day we are proving it is possible to have a commercial relationship that benefits free software."

That commercial relationship still is not one that happens between the tops of the vendor and customer bureaucracies. Among big customers it happens down among the middle tiers on both sides. For MySQL, that list of customers is impressive, and it includes Nokia, Yahoo, NASA, Silicon Graphics and Cisco. Jeremy Zawody, a self-described “technical yahoo” with Yahoo Finance, says, “MySQL will penetrate the enterprise similarly to the way (Microsoft) SQL Server did, but with much greater speed. MySQL is to Oracle as Linux is to Windows. It will slowly but steadily creep up the food chain, just like Linux has.” But when I asked Mårten Mickos if MySQL is competing with Oracle yet, he said no. “We complement Oracle far more than we compete with it.”

Still, we're at a point in history where the action is clearly shifting up the stack, from operating systems and applications to data. “We think it's the information age, not the operating system age”, Larry Ellison says. “The OS manages hardware; we manage the software.”

Because more and more of that software runs on Linux, Oracle has wisely chucked its long-standing OS agnosticism and repositioned itself, alongside IBM, as one of the world's leading “Linux companies”. Wim Coekaerts (otn.oracle.com/oramag/Coekaerts.html), head of Oracle's Linux kernel team, says “Linux is really, really important to Oracle. We are very much a Linux company.” He proudly credits the work his kernel development team has contributed to helping to make Linux “enterprise class”. There is proof in the customer pudding, too. Roland Smith says:

> I would say that Oracle is probably the best vendor in the marketplace, in respect to Linux. When we told our Oracle account team that we wanted to put up an Oracle database on Linux, they were all over it. They really knew what they were doing. Within a week they had us in touch with their development folks back in Redwood Shores. They had good documentation. It was easy to put up. They were patient. It was easy to run, easy to connect to. We're very happy about it.

Smart companies naturally want smart relationships. Oracle seems to be doing a good job of meeting that market demand. It should help them continue to adapt to the successes of MySQL and PostgreSQL.

The New Standard Story

Like Oracle, other vendors will need to adapt to a world where Linux and its open-source companions serve as fundamental infrastructure for IT. That infrastructure is quickly becoming as standard as two-by-fours, ten-penny nails and sheetrock screws. Vendors always will be welcome to take advantage of that infrastructure and to contribute to its improvement; but their frame of reference will shift from the abstract to the concrete—from abstract playing

fields to concrete IT projects where they have something useful to contribute. When that happens, and the software industry finishes growing up, credit will finally go where it's long overdue: to the smart people who used Linux to make their companies smarter, no matter what those companies bought and sold.

Doc Searls (info@linuxjournal.com) is senior editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Free Beer Doesn't Sell

Ethan Zuckerman

Issue #111, July 2003

In countries where the main competition is a cheap illegal copy, transparency and customizability are the real reasons to choose Linux.

Open-source software is customizable, expandable and available at low or no cost. Closed-source software is difficult to customize or expand and often is exorbitantly expensive. As a result, businesses and governments worldwide are beginning to make extensive use of open-source technologies, in some cases going as far as introducing legislation to mandate its use. You'd expect developing nations to be leading this charge for free, high-quality software, but you'd be wrong.

In Geekcorps' experience supporting over a hundred IT projects in developing nations, open source is a surprisingly difficult sell. The chief mistake in marketing to the developing world is an overemphasis on “free beer”. The expression comes from Richard Stallman's demand that software be “free as in 'free speech', not free as in 'free beer'”. Emphasizing the low cost of open-source software often backfires. The history of technology transfer and appropriate technology projects includes efforts that dumped obsolete technology on developing nations when it was no longer marketable in developed nations. As a result, inexpensive and inferior are often falsely equated.

In fact, free software frequently costs users in developing nations more than proprietary software. Widespread copyright infringement means software often is available for the price of the media on which it's delivered. In Yerevan, Armenia, I recently found Microsoft Windows XP and Red Hat 8 shelved side by side, both selling for less than $5 US. For users familiar with Windows, Linux has an incremental cost—the cost of the manuals necessary to use the software. And, at copyright infringement prices, manuals can cost 20 times as much as software.

Although on-line documentation can address some of these problems, it can be expensive and intimidating to access this information. Downloading the 3MB Emacs manual gets expensive when you're paying by the minute for connectivity at a cyber café where 20 machines share a single 28.8kbps modem. Asking questions of experts in another country, often in an unfamiliar language, becomes even more intimidating when new users encounter netiquette for the first time. A well-meaning response of “RTFM” is likely to be misinterpreted by someone from a culture that places a high value on politeness or formality. Almost every culture in the world is more polite than global geek culture.

Although open-source advocates need to consider the linguistic, cultural and bandwidth barriers that affect adoption in developing nations, our chief focus must be on demonstrating that open-source software can be more advanced and powerful than the alternatives. This requires a focus on the free speech benefits of open source: expandability, transparency and resulting high performance. A great example of “free speech” application development is the translate.org.za project, dedicated to providing operating systems and critical applications in all 11 of South Africa's official languages. Thanks to their work—and the availability of source code for Mozilla—a browser now exists in Zulu, Xhosa and four other languages.

IDN, one of Ghana's leading ISPs (www.idngh.com), is refocusing its business around its SAVA wireless access points. Manufactured in Ghana, SAVA boxes are Linux servers that let businesses and cyber cafés offer connectivity using WiFi, avoiding Ghana's inadequate local phone system. ISPs around the continent are looking at IDN's success, and some are migrating their architectures to Linux servers.

Before open source can have a true worldwide audience, developers need to focus on IT problems specific to developing nations. Efforts such as Simputer (www.simputer.org)--a low-cost Linux-based handheld for India—suggest one model for these efforts: a cadre of talented engineers from a developing nation supported by a global community. MIT Media Lab's ThinkCycle Project (www.thinkcycle.org) demonstrates another model, matching design students in universities around the world with the engineering challenges of nonprofit organizations in developing nations. The US government's new Digital Freedom Initiative (www.digitalfreedom.gov) is trying a third model, creating solution teams of Senegalese developers, business and IT expert volunteers from US and Senegalese end users, to create new IT applications for use in Senegal and throughout Africa.

Political developments also may advance the global adoption of open source. An increasing number of politicians, like Peru's Edgar Villanueva Nuñez, are

pushing legislation to mandate the use of free software to protect the “integrity, confidentiality and accessibility” of information.

Advocates in the developed world can push in a different direction. Demand that money spent by your nation's foreign aid agency on development of new IT products in developing nations goes toward the creation of open code. The result likely would be a boom of developers in poor nations and a wealth of code that could be used in other developing nations—not to mention a few more Xhosa-speaking geeks.

Ethan Zuckerman is the founder of Geekcorps (www.geekcorps.org), a nonprofit volunteer organization dedicated to helping developing nations achieve digital independence by building successful IT businesses. He is a fellow at the Berkman Center for Internet and Society at Harvard Law School (cyber.law.harvard.edu). You can reach him at ethan@geekcorps.org.

Archive Index Issue Table of Contents

Advanced search

Astaro Security Linux V4

Jeremy Impson

Issue #111, July 2003

Astaro delivers an impressive array of security features in one package.

Astaro Security Linux (ASL) V4 is a nifty all-in-one firewall product. Billed as an affordable enterprise-level security solution, Astaro delivers an impressive array of security features in one package. ASL V4 is a software product, not an appliance, so you need to provide your own hardware for it.

The ASL feature set leaves little to be desired. It uses the journaling features of the ext3 filesystem for rapid recovery from unplanned reboots. All administration is performed from an SSL web front end. It also comes with an SSH server that provides command-line access, a fact that pleases this reviewer. Astaro does state that SSH access is intended only for advanced users, because users would have to edit the configuration files by hand. Still, I was pleased to have the option of getting down and dirty with the system if necessary.

ASL performs packet filtering, just as you'd expect a Linux- and iptables-based system would. Source NAT, destination NAT and IP masquerade are all supported. The default ruleset is quite locked down and is modified automatically when you add or modify services. But ASL adds further value with its use of network groups. ASL has a single set of user interfaces to let you define networks, network group, service groups and user objects. Once defined, ASL lets you use them for all sorts of access control lists (ACLs). For example, once a network group is defined, it can be used when configuring NTP services and defining IP-masquerading rules. Another example can be seen with the predefined service group called netbios, defined as netbios-dgm (port 138), netbios-ns (port 137) and netbios-ssn (port 139). With this group, one can set various rules that consistently affect all three of these services.

As with packet filtering, ASL comes with the various proxy packages we expect. This includes an HTTP cache that can be configured to work transparently to

the client, SOCKS proxy and SMTP relay. The SMTP relay includes antispam and virus scanning subsystems. ASL V4 also includes IPSec, DHCP, PPTP, identd, POP3 with virus scanner and DNS forwarding services. Service includes a virus scanning subsystem.

For all the proxies and services that can make use of user accounts, such as SOCKS, HTTP and SMTP, ASL can use your existing LDAP, RADIUS or Windows Domains to provide authentication, or it can use its own user account database. Once identity has been established, access controls can be placed on users to control access to different services.

ASL has a port-scan detection system that sends e-mail to an administrator when it detects port-scan activity. It also can be configured to drop or reject the port-scan traffic. Interestingly, the source of the port-scan traffic is allowed to communicate again after the port-scan traffic has ceased.

Astaro has incorporated an automatic update function into ASL, called Up2Date. (It isn't clear what relation, if any, this software has with Red Hat's up2date update tool.) All patches are signed digitally, and all communications are encrypted. Besides patches, virus and spam signatures also are updated with this mechanism. Updates can be made manually—scheduled hourly, daily or weekly—or can be downloaded by hand from Astaro's FTP server. System configuration backups can be exported either manually or automatically. These backups can be encrypted and e-mailed.

ASL offers load balancing, using iptables NAT rules to split incoming connections between two or more actual servers. ASL seems to be protocol-agnostic; that is, it performs load balancing for any service protocol. Naturally, it remains for system administrators to ensure that the servers sharing the load are able to synchronize themselves if necessary.

Even the user guide has value beyond explaining how to use ASL. It provides a short but accurate tutorial of the various technologies involved in network security. For example, it discusses transparent vs. application proxies, and it contrasts them to IP masquerading.

ASL really has too many features to discuss fully in this review, including syslog support, wireless LAN access point mode, WEP support, the capability to plug in to an untagged port of a VLANed switch (why this can be beneficial could be the subject of a whole other article), a network time protocol client, high availability and quality of service.

You can configure the system logs for authorization, dæmons, kernel, notifications and SMTP relay messages to be logged independently.

Astaro recommends a minimum system configuration of a single Intel Pentium II (or equivalent) processor, 128MB of RAM, an 8GB disk drive (ASL supports IDE and many popular SCSI cards) and two or more PCI network cards. If you wish to use wireless network PCMCIA cards, you should use one based on the Prism2 chipset. Certain other configurations, such as high availability or the VLAN subsystems, may require other specialized hardware.

The installation happens with a text/curses-based menu and is fairly quick and simple. It asks a few questions and then proceeds to repartition and reformat the hard drive, so be sure you're ready to let that happen.

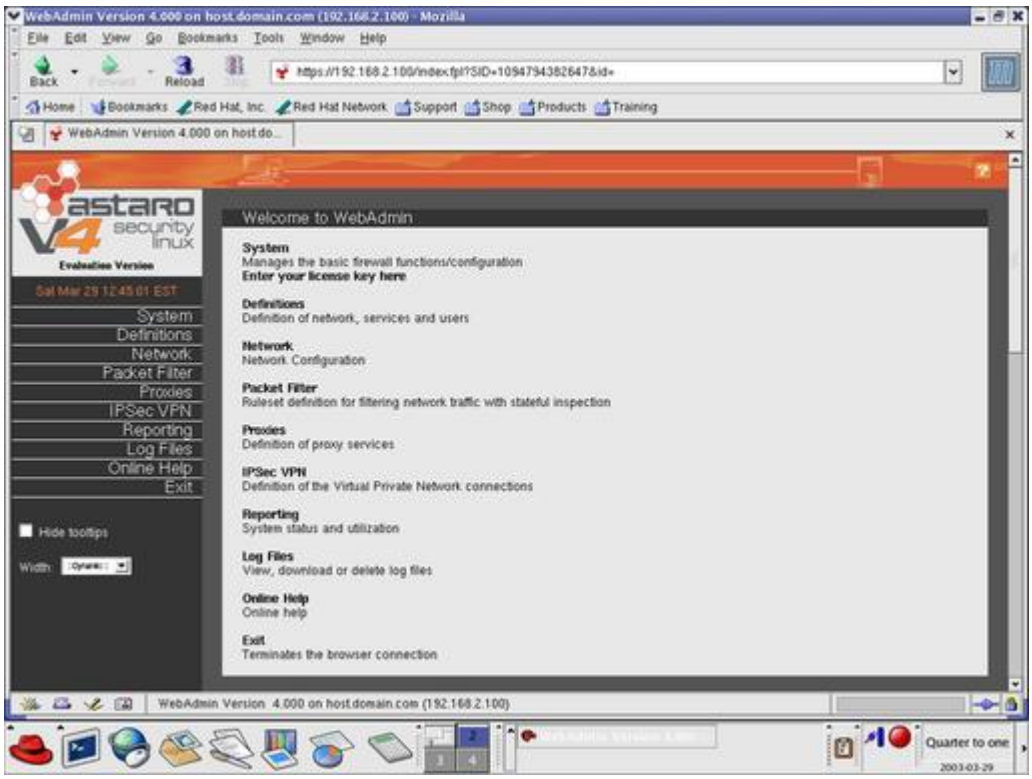


Figure 1. Manage Astaro Security Linux with an SSL Web Tool

Once installed, you point a web browser at a secure web GUI. The best adjective to describe the look of ASL's web GUI is slick; it is well designed and has some impressive functions. One function allows you to kick off other administrators, with the option of providing them a reason for the boot. Logged-in sessions have an automatic timeout as well. There's easy access to the system's uptime, last login, date and time.

ASL has a nice iptables/firewall display, with GUI interfaces to add your own rules, and easy-to-understand ICMP settings, including options to make firewall visible to traceroute, make firewall forward traceroute, make firewall ping visible and make firewall forward pings and so on. This reviewer was most impressed with the live log of filtered packets, whose regularly updating display shows recent network activity. The only thing this display lacks is some way to

know which firewall rule caused a packet to be denied. Such information is essential for rule debugging.

ASL's slickness extends to the SMTP GUI and its options for access control and many ways to try to fight spam, such as verify address, black holes, file type filter and expression filter. If you've ever tried to configure sendmail, you know what a daunting task this can be.

Besides configuration, the web GUI also contains a number of graphs. The graphs plot statistics for CPU, RAM and swap; current hard-disk usage and process list; total network connections; and each network interface for the past 24 hours.

ASL has a few other remarkable features. The first is the IPSec interface has a certificate management option. It can generate or upload its own Certificate Authority (CA), and it can receive and even fulfill requests for new client certificates. If your organization does not have a certificate authority, ASL can provide its own CA key material. Remember, a whole slew of policies and procedures should be defined before one could consider one's CA actually to be secure.

An ASL machine beeps five times at boot and shutdown, letting you know its status with certainty. Most of the system services run in a chroot()ed environment. Some even have their own disk partition, which helps to prevent disruption of other services that might otherwise result from a denial-of-service attack on one service. The administrator can install custom software. ASL could be a nice starting point for your own Linux-based projects.

ASL is not a typical home-use broadband firewall-router. It certainly can be used for that, but because ASL can do so much more than DHCP and IP masquerading (or NAT, if you prefer), typical home use would be a waste of its features. Out of the box it requires some setup, but if you already know you need more than a typical broadband firewall-router appliance, chances are you already know how to achieve this. ASL would make a good home-use firewall-router for the network-savvy power user, someone comfortable with or interested in learning the ins and outs of SMTP configuration, SSH keys, RADIUS configuration and/or IPSec. So it's fortuitous that Astaro provides a free home-use license, which does not include the antivirus protection.

I do have a few minor reservations about ASL V4. I admit that ASL impressed me with the sheer number of features it supports. It is common security policy to limit the functional responsibilities of each infrastructure node on your network. For instance, a router should stick to routing, a firewall should stick to packet filtering, a proxy server should stick to application protocol proxies and

an SMTP relay should stick to e-mail. The justification is that a separation of functions limits the overall damage should any one service suffer a security exploit. With everything in one box, ASL might encourage an all-in-one approach. On the other hand, such a configuration may be desired if there are cost, space or administration constraints or, simply, if the risks aren't as high. Indeed, there's no reason one couldn't install ASL on numerous machines and turn on only certain services in each node. But this illustrates another potential problem. Aside from the ability to export and import configurations, there is no apparent support for a single point of administration for multiple ASL nodes. In other words, there's no policy manager that can push configurations to multiple ASLs.

At installation time, you receive a 30-day license, after which you need to purchase a full license. At the time of this writing, prices range from $390 US for a ten-IP network and a one-year Up2Date subscription to $6,895 US for unlimited IPs with three years of Up2Date. Virus protection, high availability and surf protection are priced separately. Astaro has a free personal use license, and this reviewer intends to take advantage of it.

Product Information

Jeremy Impson is a security and network consultant in Upstate New York. Send your questions and comments to jdimpson@acm.org.

Archive Index Issue Table of Contents

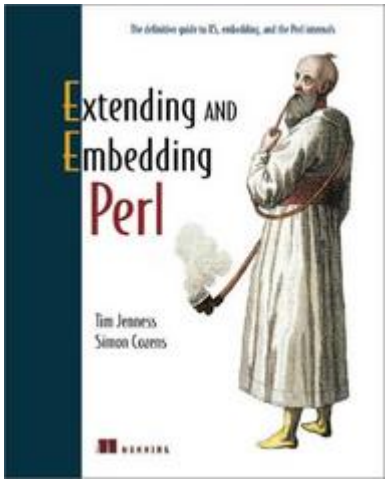
Advanced search

Extending and Embedding Perl

Paul Barry

Issue #111, July 2003

Extending and Embedding Perl by Tim Jenness and Simon Cozens



Manning Publications, 2003

www.manning.com/jenness

ISBN: 1-930110-82-0

$44.95 US

Nothing instills more fear in the average Perl programmer than working with XS, Perl's technology for interfacing with eXternal Subroutines in other languages, most notably C. Calling C libraries from Perl is a delicate science, as is the related technique of embedding the entire Perl interpreter into an existing C program. Not every Perl programmer needs to do such things, but when faced with the task, it's hard work.

Written by two respected members of the Perl community, *Extending and Embedding Perl* attempts to address the needs of such programmers. It covers

a lot of ground: XS, the Perl API, alternatives to XS, embedding, the Perl internals, compiling and the Perl development process.

I particularly liked the use of real examples (and working source code) when discussing various aspects of this technology. There are code snippets from Tk, Apache::mod_perl and the Perl sources. I also liked how the authors often would present an obvious solution to a problem, only to highlight its shortcomings, refine the initial solution and then present the improvement. This made the book's material real and useful.

I don't agree with the authors' assumption that they provide enough C in the two chapters they devote to the language. Programmers already competent in Perl *and* C will get the most from this book. If all you know is Perl, you'll likely struggle with the material. This is not really the authors' fault: XS isn't easy, and the style of C that's used within the Perl sources and the XS interface is somewhat obscure (that's being kind). I was amazed how often I found myself looking at the presented C code and shuddering. That aside, this book is a welcome addition to the arsenal of Perl books. It should help with demystifying XS and associated technologies. As to whether *Extending and Embedding Perl* is a book for every Perl programmer, I'd have to say it is not. However, if you do need it, you will find it an invaluable reference.

—Paul Barry

email: paul.barry@itcarlow.ie

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #111, July 2003

Readers sound off.

Best Cross-Platform GUI Toolkit?

I am writing a printed-circuit-board layout program called FreePCB, which I intend to publish on the Internet as an open-source project. I chose to write it for Microsoft Windows, and if it is successful I would like to port it to Linux. When it comes time to port it to Linux, how would you suggest that I proceed?

—Allan Wright

One way to do a cross-platform application is with wxWindows (see page 90). It's being used in the upcoming Chandler cross-platform mail and calendar—Ed.

Sales Departments Need Help

I have been reading a lot about failed distributions lately. I would like to recommend that sales departments take a close look at how they handle customer relations. Mandrake Linux refuses even to post an e-mail address for pre-sales questions. I have attempted to contact these people with detailed questions about product component level support, prior to purchase, without success. Red Hat refuses to respond to the individual user for pre-sales inquiries. Heck, I have even tried to get information out of them for my place of employment, an enterprise-level situation. SuSE—my hat is off to these guys. You may not get the perfect response, but they do acknowledge you exist and try to help. I have a copy of 7.3 Pro and am most likely to purchase another version once I get a home wireless network. Debian—again, my hat is off, and they don't even sell anything. I have had many responses to inquiries from their support base. Thanks to SuSE and Debian. Please keep up the good work.

—John R. Klaus

alt.fan.robert-love

I want to thank you for getting Robert Love to write the article on kernel 2.6. The article was simply superb and explained many facts about what happens during the kernel development process.

—Ravi

Vive le Chef!

I give credit to Mr Gagné for my first purchase of *LJ*. When teaching at a local college, I directed my students to articles of his that were relevant to topics covered. I joined his site mailing list and purchased his book because of his style of writing. Your web site has given me the opportunity to publish articles I have written, and I will admit my writing style tends to mirror Mr Gagné's. I find a lighter writing style mixed with a human element has character. Computer concepts can be brought to life. It is our responsibility as authors to make it happen. I have strong memories of teachers and writers who moved off the mainstream path to deliver their message. You are doing something right when you offer Marcel Gagné's articles to your readers.

—Sean D. Conway

Spam Kills

Twenty billion junk e-mail messages sent per day may potentially take 20 billion seconds to delete. A human life is a mere two billion seconds long. In effect, spammers kill ten people each day. If *Linux Journal* financially supports a business that offers web hosting services to spammers, then *Linux Journal* in effect backs spamming. It would be more appropriate for *Linux Journal* to question the allegations that Rackspace harbors spammers, than to question the need of the Internet community to take meaningful action against network abuse.

—Anders Andersson, Uppsala University

Shirt Sandbags Linux at Church

Linux Journal's silly “Linux Saves” T-shirt really undermines my effort to recommend the use of Linux in the United Methodist Church. I feel that churches would greatly benefit from Linux and other open-source software, and I don't understand why someone would create a product like this. Maybe they thought it would be funny outside the church, but inside, nobody is laughing.

—Mark Ramsell

Wine Recommendation

Please, please keep Marcel and his trusted assistant, François exactly as they are! Not only does it help the Francophiles among us to brush up on our French, but the levity it provides actually aids the cognitive process, at least in this reader's humble opinion. Marcel, when are you going to feature a fine Virginia wine?

—Pat Murphy

Red-Eye No More

I fixed one of my digital images after reading Eric Jeschke's excellent tutorial in *LJ*'s April issue, and I wanted to thank you for this excellent series. I can't wait to read Eric's next article.

—Mike

Out for a Drive

I thought it might interest you to see how high your adverts get in South Africa. Somewhere in that vehicle is my Dell Inspiron 2600 running Slackware 8.1 and GIMP 1.2.3, which helped to make this image in conjunction with my Fuji MX2700 digital camera.

—Alf Stockton

Resting at Sani Top after going down and up Sani Pass on 22nd March 2003

Yay Kernel, Boo Pixelated Games

I just got my *Linux Journal* (May 2003) in the mail the other day, and many pressing things were put aside while I read it. The article on the 2.6 kernel typifies what I like about your magazine. Heather Mead's column, “Adaptability and Ingenuity” left me a bit confused. Am I the only one who doesn't know what a theremin is? The Upfront section now and again features programs so old-school they make me cringe; your review and picture of Football Manager is a prime example. The graphics are nothing if not from Atari 2600. These sorts of crufty-looking programs just don't cut it.

—Matt Wyczalkowski

The theremin was the first electronic instrument, invented by Leon Theremin in the 1920s and still played today. And, some of the oldest-looking games have the best play value—Ed.

Another Linux License Plate

Here is another license plate for you. “LINUX” was already spoken for in my province, so I chose to describe myself rather than the OS.

—Richard Weait

Fresh Cuisine

My wife and I are absolutely pleased with the fresh, unique articles written by Marcel Gagné. On seeing a negative comment about the French chef's approach in the May 2003 Letters department, I just had to write and defend that which we enjoy so much.

—Glen Farmer

Erratum

On page 12 of the April 2003 issue, the quotation by Craig Sanders should have been attributed to an interview done by Sam Varghese, which was published on

the technology web sites of *The Age Online* and the *Sydney Morning Herald Online*.

Archive Index Issue Table of Contents

Advanced search

Upfront

Various

Issue #111, July 2003

They Said It, diff -u and more.

BitTorrent: bitconjurer.org

Want to share your GNU/Linux ISOs or your 950MB FLAC recordings of your favorite taping-friendly band? With Bram Cohen's BitTorrent, you can serve big files to many users at once, even from a dial-up or DSL connection, by making the clients do most of the work. As soon as any BitTorrent client has part of the file, it helps serve that part to others. BitTorrent installs easily as a helper application in Mozilla or other popular browsers. Requires Python.

—Don Marti

Crossfire: crossfire.real-time.com

If you like role-playing games, look no further. *Crossfire* is a multiplayer magical role-playing game that will keep you from getting any constructive work done for weeks, maybe longer. You can play on servers all around the world or run your own server. It is definitely a lot more fun playing cooperatively in a small party than playing alone. But, do remember to eat and sleep occasionally. Requires: X client: libpng12, libz, libm, libX11, libXext, glibc, libdl; Gtk client: libpng, libz, libm, libgtk, libgdk, libgmodule, libglib, libdl, libXi, libXext, libX11, libSDL, libpthread, glibc.

—David A. Bandel

diff -u: What's New in Kernel Development

New kernel driver components are cropping up to take advantage of modern high-speed serial ports. **SuperIO** chips have supported up to 460.8K and 921.6K baud for a long time, though no Linux driver existed. In February 2003, **David**

Woodhouse posted a patch to give Linux access to the high-speed capabilities in those chips.

It's now possible to use thousands of hard drives under Linux. The previous limit was 256. **Badari Pulavarty** submitted a patch in March 2003 to raise that limit drastically. During one test run, he successfully read from and wrote to 4,000 distinct disks. The new limitations on the number of disks supported by Linux still is being investigated. Various resources become overburdened as the number of disks rises, memory usage being the main catchall for these problems. Ways of accommodating the RAM requirements of massive numbers of disks are being investigated actively. A true hard limit on the number of disks may re-emerge eventually, but not until various debris has been cleared away.

A new GPLed graphical bootloader, called **Gujin** (short for GPL Use of the Jnp INstruction), emerged in April 2003, billing itself as a replacement for **LILO**. Coded entirely from scratch by **Etienne Lorrain**, Gujin boasts the ability to identify available kernels at boot time, which is an improvement over LILO's need to modify the boot sector for each installed kernel; the ability to load extremely large kernels, because LILO's size restrictions are being strained by the growing size of recent 2.5 kernels; and a sexy graphical interface for choosing which kernel to boot. It's still quite a new project, so don't look for it to take over the world in the immediate future; however, it does promise to be a flexible, powerful alternative to LILO and **GRUB**.

A new set of API functions for **USB** gadgets has been produced by **David Brownell**. This is intended to be used by peripheral devices, such as PDAs and other embedded systems running Linux. The API functions provide a consistent interface so that software written for one peripheral device can run unchanged on another, in spite of the fact that the two pieces of hardware may be quite different.

Now you can track the input/output readiness of files in a portable manner, using the new **libivykis** library by **Lennert Buytenhek**. This past April he released the library as a wrapper around functions like poll() and kqueue(), enabling a standard interface for high-performance networking servers, regardless of the system on which they happen to be running. The **SourceForge** page for libivykis lists it as fully stable and ready for production use, in spite of the fact that the first public release took place only recently. Whatever the case, libivykis apparently has been used already to produce proxy servers for various protocols, as well as a streaming video server.

The proprietary **BitKeeper** (BK) tool is becoming more and more firmly entrenched as the version control system used in kernel development. A new real-time BitKeeper-to-CVS gateway, set up by BitKeeper's creator, **Larry McVoy**,

has silenced some of the more vocal anti-BK kernel developers. At the same time, free alternatives like **arch** and Subversion appear to be banding together, working to solve the formidable problems that have kept free version control systems from providing the state-of-the-art features found in BitKeeper.

—Zack Brown

Garbage Subtractor

Students at MIT's Laboratory for Computer Science (www.lcs.mit.edu/news/harddrives.html) recently looked into 158 disk drives obtained from eBay. Among other data, they found 5,000 credit-card numbers, numerous medical records and gigabytes of personal e-mail.

Graduate student researchers found a “widespread lack of awareness about how to remove sensitive data from hard drives effectively before recycling or discarding them.”

Phil Howard created a simple Linux-based solution to the problem: DiskZapper, a bootable Linux distribution, wipes all drives connected to the computer (diskzapper.com). It's free software, or you can order it on floppy or CD-ROM. Proceeds go to LinuxLobby.org.

—Doc Searls

NYFairUse Protests EGOVOS Conference

New York City's fair use activists, costumed as the American Founding Fathers, left the warm comfort of their homes at 4:00**AM** on March 17, 2003, to attend the EGOVOS conference at George Washington University in Washington, DC.

The conference was supposed to be a showcase for free and open-source software in government. Instead, organizer Tony Stanco turned it into a platform and photo opportunity for Microsoft. Microsoft threatens free speech and fair use by taking control away from users with Digital Rights Management and with its upcoming Palladium “trusted computing” platform.

We are fortunate to have associates working on Broadway, and they introduced us to costume designers who dressed us as patriots of 1776. At the conference, I personally had the pleasure of speaking about the problem with European Union Minister Philip Aigrain, whom I met in Bordeaux last year.

After our trip to George Washington University, we made a trip to Capitol Hill while still dressed in our costumes. We got big smiles all along the halls of Congress, especially at Congressman Weiner's office. He's a member of the

subcommittee on Intellectual Property and the Internet. We have a handshake deal to install a GNU/Linux system in his office, so stay tuned.

—Ruben Safir

Top row: Adam Kosmin (Windows Refund Day), Ray Connolly (Free Software Chamber of Commerce Chair) and Tim Wilcox (CEO of Linux Force Beowulf Cluster Expert). Bottom row: Ceasar Vargus (NYLXS Member), Sunny Dubey (Poly Tech Student and NYC Board of Ed IT Manager) and Ruben Safir. All are members of New Yorkers for Fair Use (fairuse.nylxs.com).

LJ Index—July 2003

1. Height in centimeters of Wakamuru, a Linux-based Japanese robot designed to care for the sick and elderly: 100
2. Size in words of Wakamuru's vocabulary: 10,000
3. Percentage of IT budgets spent on maintaining existing systems: 79
4. Percentage range of Linux cost savings relative to UNIX, running Oracle: 45-80
5. Billions of yen Japan's Ministry of Economy, Trade and Industry will dole out for open-source development in the next year (2003): 1
6. Millions of yen budgeted in the next fiscal year to study possibly switching Japan's government computers to open source: 50
7. Cost in US dollars of an SGI mainframe replaced by a cluster of 12 PCs running Red Hat Linux at the Johnson Space Center in Houston: 1,600,000
8. Annual maintenance costs in US dollars of the SGI mainframe: 50,000
9. Cost in US dollars of the 12 PCs and their OSes: 25,000
10. Number of the top ten hosting locations that run Linux: 9

11. Percentage drop in IT spending during 2002, according to IDC: 4.1
12. Percentage of companies planning to increase IT spending in 2003, according to IDC: 85
13. Percentage decline in average business spending on computer hardware and software in 2003, according to Goldman Sachs: 1
14. Percentage increase in IT spending forecast for 2003, according to Forrester Research: 1.9

Sources

1, 2: Linux Devices3, 4: Oracle5, 6: Associated Press7-9: Information Week10: Netcraft11, 12: International Data Corp.13: Golman Sachs14: Forrester Research

In Memoriam

Mike Jackson, Linux Shadow Password HOWTO author, passed away on Friday, March 28 at the age of 38. In addition to maintaining the LSPH all these years, Mike was also a Navy sonar instructor, Reserve police officer, president and

founding member of TSCNet, treasurer and founding member of the Kitsap Peninsula Linux User Group and contributor to projects such as the PHP SNMP module and the Sharp Camera package for OpenZaurus.

The last update of the Linux Shadow Password HOWTO was in 1998. This isn't because Mike lost interest, but because the HOWTO convinced everyone to start including shadow password support in their distribution by default.

When Mike took on a new project, he dove in headfirst. When he found bugs, he'd fix them and contribute the patches back to the project. When he found a lack of some feature, he would add it himself. He set an example for what a true hacker ought to be, demanding excellence and openness from software and improving it where there were flaws.

Mike's passing has deeply affected all of his family and friends; we will miss him greatly. We have gathered together the pieces of his home page and reassembled it along with other links at www.kplug.org/~mhjack. If you have memories of Mike you would like to share, please e-mail them to bcl@brianlane.com, and I'll add them to his home page.

—Brian C. Lane

They Said It

Universities are very cost-conscious, and the relatively low expense of Intel systems combined with organic availability of Linux is quickly becoming the norm, especially at reputable engineering schools. The free availability of the source code leads universities to standardize on Linux for computer science courses.

—Billy Marshall, Red Hat

Continuing core development of version 3 is mostly paid for by Hans Reiser from money made selling licenses in addition to the GPL to companies who don't want it known that they use ReiserFS as a foundation for their proprietary product. And my lawyer asked “People pay you money for this?” Yup. Hee Hee. Life is good. If you buy ReiserFS, you can focus on your value add rather than reinventing an entire FS.

—mkreiserfs, in reiserfsprogs 3.6.5

The first discovery I'd like to present here is an algorithm for lazy evaluation of research papers. Just write whatever you want and don't cite any previous work, and indignant readers will send you references to all the papers you should have cited.

—Paul Graham

Comparing UNIX and Linux like-for-like, we found that we get two to five times the amount of throughput [messages per second] on one of the Intel boxes than on a Sun Sparc box, at half the cost.

—Casey Merkey, Global Linux Program Manager, Reuters' Market Data System

—David A. Bandel

trickle: monkey.org/~marius/trickle

I often have a low-priority task that takes some time running in the background, like a large FTP download. With trickle, I can manage bandwidth usage on a per-program, per-IP-address basis, so my SSH sessions are still responsive, my FTP sessions continue (albeit at a slower pace), and family members and coworkers don't get upset. Its only drawback is that you must remember to use trickle to invoke the program for which you want the traffic shaped. Requires: libevent, libnsl, libdl, glibc.

—David A. Bandel

Archive Index Issue Table of Contents

Advanced search

Fun with Hardware

Don Marti

Issue #111, July 2003

From 64-bit servers to console conversion projects, your Linux platform choices are better than ever.

From the Editor

Fun with Hardware

From 64-bit servers to console conversion projects, your Linux platform choices are better than ever.

by Don Marti

Running Linux makes you smarter, and we've got proof. In his article on Nagios on page 52, Richard C. Harlan explains how John Deere integrated its diverse server management needs under the thumb of one Linux-based project, for a small budget.

Other cluetrain-riding people at a variety of companies talked to Doc Searls about the new balance of innovation power between informed Linux-using customers and their vendors (page 38). Freedom is changing companies behind the scenes, and thanks to your discreet tips, Doc is watching it better than anyone.

This issue also includes the year's best Linux hardware news so far. The big cheeses of the information technology industry are building servers based on AMD's new AMD64 architecture, which you may know as **linux/arch/x86_64**. In an interview on tazaa.info, AMD CEO Hector Ruiz included Linux as one of only two “operating systems that matter”, and our favorite OS was the first one released for AMD64.

You can get an idea of AMD64's abilities, and those of the Newisys two-way server that's among the first Opteron products on the market, in Michael Baxter's first look on page 58. Michael is the man to ask about Linux in the electronic design automation industry, and the new AMD architecture is already attracting attention from Cadence and others as a way to replace expensive 64-bit RISC UNIX.

Our other featured hardware article this issue covers Microsoft's Xbox video game system. On page 44, Michael Steil explains how making the Xbox run Linux is not only fun but actually useful. Try it. Upgrading an Xbox is a great way to learn about the boot process and is cheaper than a single-board computer for hobbyist embedded projects. Cut back on the coffee the day you solder those two little pads together, though.

If your web site uses free software exclusively, you might not realize how big of a deal CMF for Zope (page 14) really is. Proprietary content management systems have high-priced licenses and *still* require you to do substantial customizing. This might be the article that makes you a web hero at work, so pay attention.

This issue also hosts a cross-platform development-tool cage bout. Will your next project use the promising new Mono (page 74) or the reliable wxWindows (page 90)? Both are free as in Dmitry, so you can easily try both. Finally, contributor Josh Rabinowitz told me that he got hooked on using the man page index, shown in his article “How to Index Anything” (page 82), before he was even done with the article. Imagine searching all your man pages, *Linux Journal* archive CDs and old mail with one tool. I'm going to try it out.

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

Archive Index Issue Table of Contents

Advanced search

Taking Matters into Our Own Hands

Heather Mead

Issue #111, July 2003

Waiting around for vendors to give us what we want usually means waiting a long time. Besides, it's just not our style.

On the Web

The fact that many items on your tech wish list have yet to find their way to Best Buy doesn't mean you can't find a workaround. One of the biggest benefits of using open-source tools is not having to wait for other people to give you what you want. While Doc Searls and others keep their eyes on the big vendors and suppliers, wondering when the customer will gain the upper hand, our web authors have been supplying us with articles about what we can do for ourselves in the meantime.

Take, for example, Linux on the laptop. Other than the recent availability of the Lindows MobilePC (www.linuxjournal.com/article/6662), the options for using Linux on one's laptop have been limited, to say the least. Between system incompatibilities and lack of support, getting Linux up and running may seem next to impossible for many. To this end, Jay Docherty has been writing a web series for us that takes people from the purchasing phase to setting up GNOME, selecting themes and enabling sound. “Configuring Your Laptop for GNOME and Sound” (www.linuxjournal.com/article/6809) explains how to install GNOME 2.2 for Debian, configure support for the i810 chipset, change themes and set up a USB mouse on the laptop.

Another example of assuming responsibility for one's own needs is Roberto de Leo's article, “Self-Hosting Movies with MoviX” (www.linuxjournal.com/article/6474). de Leo wanted to find a “Linux CD mini-distribution that is able to boot and play automatically all audio/video files on the CD”. His internet search for such a CD proved fruitless, so he decided to build it himself. The result is MoviX, which fulfilled his need for a dedicated CD mini-distribution. His step-by-step

creation article, however, can be adapted for whatever purpose your mini-distribution might serve.

The quest to have the coolest, fastest, quietest and, simply, ultimate Linux box leaves one with no option other than to build it oneself. Without a doubt, one of *Linux Journal*'s most popular yearly features is the Ultimate Linux Box (ULB) article, where we lay out the components of our dream machine and the reasons why it's so dreamy. This year, we're doing things a little differently in the spirit of community collaboration. We're using the web site as a launching pad for the ULB conception. Articles will be posted naming the contenders in each area. Glenn Stone started the discussion with “Ultimate Linux Box: a Case Study”, (www.linuxjournal.com/article/6764). We're asking for your input, in the form of e-mail to the authors and postings on the article web page, to help make the final decisions. Don't wait until the final ULB is unveiled to tell us where our mistakes are; speak up now.

If you have taken matters into your hands to get what you want for Linux, send article ideas to info@linuxjournal.com. Be sure to visit the *LJ* web site often; new articles are posted every day.

Heather Mead is senior editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #111, July 2003

Our experts answer your technical questions.

Best of Technical Support

Kickstart from USB Floppy?

I am using the Red Hat Kickstart installation feature for the 7.2 distribution to install a customized version of Linux on several machines. Currently, I use a floppy drive to boot and store the ks.cfg file on the floppy, so the syslinux.cfg file has a line that looks like this:

```
label ks
kernel vmlinuz
append text ks=floppy initrd.img lang=
devfs=nomount ramdisk_size=7168
```

How do I start the Kickstart installation using a USB floppy/USB CD-ROM? I can boot using USB floppy/CD-ROM, but the Kickstart installation fails, as it doesn't find the ks.cfg file residing on the USB device.

—Vishali Karnik, Vishali.Karnik@respironics.com

According to their release notes, Red Hat did begin recognizing USB floppy drives during install with the 7.2 release, so this should be possible. I don't have a USB floppy drive to test this, but the helpful people at Fujitsu Siemens Computers have some advice on how to do Kickstart from USB floppies: www.fujitsu-siemens.com/partner/linux/readme/driver-disks-redhat.shtm. USB floppy drives are detected as SCSI devices. If you have no real SCSI devices on the system, you need to change **ks=floppy** to **ks=hd:sda/ks.cfg**. If that doesn't work, drop to a shell during a manual install and **cat /proc/scsi/scsi** to see what device name in /dev is being assigned to the USB floppy. The first device in /proc/scsi/scsi will be sda, the second will be sdb and so on.

—Don Marti, info@linuxjournal.com

Using a Nonstandard Modem

The PCtel 2304 WT modem with my Dell notebook is not detected by Red Hat 8.0.

—Hari Babu Prasad, hari_bsnl@rediffmail.com

In order to use PCtel-based modems under Linux, you must use a driver module. An unofficial home page, linmodems.technion.ac.il/pctel-linux, provides the latest version, a list of supported modems and a pretty good HOWTO.

—Mario Bittencourt, mneto@argo.com.br

I Have No POP and I Must Get My Mail

I just installed an internet server (Red Hat 7.3), and when I try to access the POP3 mail from another computer, it says the connection is refused. I already checked sendmail, and it's running.

—Fausto Garcia, faustog@gesnet.com.mx

There are two halves to a complete mail server configuration, and sendmail provides only one-half: the mail transfer agent (MTA) using the simple mail transfer protocol (SMTP). This is a push mechanism used for delivery of a message to a target system. It does not provide services for clients to pull messages from their mailboxes.

—Chad Robinson, crobinson@rfgonline.com

POP3 is not provided by sendmail in Red Hat; it is provided by a package named IMAP, more specifically, imap-2001a-10. To configure POP3, follow these steps: 1) load the package from rpmfind.net/linux/redhat/7.3/en/os/i386/RedHat/RPMS/imap-2001a-10.i386.rpm; 2) install it with **rpm -Uvh imap-2001a-10.i386.rpm**; 3) enable the POP3 service by editing the file /etc/xinetd.d/ipop3 and changing the line that says **disable = yes** to **disable = no**; 4) start the service with **service ipop3 start**; 5) make sure POP3 starts every time you boot your server: **chkconfig --level 345 ipop3 on**; and 6) test your POP3 service. Of course, you need to have a user account in your server.

—Felipe Barousse Boué, fbarousse@piensa.com

USB Keyboard Quits at Boot

I recently upgraded my Compaq Presario 7000 from Red Hat 7.1 to Red Hat 8.0. My USB keyboard worked fine during the whole setup process. Once the unit boots into runlevel 3 or above, however, the USB keyboard no longer works. To get around this, I have edited my modules.conf file so no USB support is ever started, which is not a great solution.

—Doug Poulin, dougp25@yahoo.com

Make sure your kernel includes "USB Human Interface Device (full HID) support" and "HID input layer support". Try **modprobe hid** to see if this actually is a module. If so, you might try adding these lines to your /etc/modules.conf:

```
alias usb uhci
post-install uhci modprobe hid
```

—Robert Connoy, rconnoy@penguincomputing.com

Linker Error

I am porting my project from Solaris to Linux and am faced with a few linking problems. The linker reports some multiple definition errors on functions. The function is defined in one .C and one .CXX file. On linking the object files created from the compilation of these objects, a multiple definition error is being issued. However, the linking step goes fine on Solaris. Is this a problem with some linker options? I am using the C++ linker on both platforms with the same flags/options.

—Mohit Kumar Singhal, mohitksinghal@rediffmail.com

Compiler/linker options are different between Solaris and Linux, even if you are using GCC on both.

—Usman S. Ansari, uansari@yahoo.com

How to Load a Module at Boot?

I'm attempting to install the Promise SuperTrack SX6000 RAID driver module into Red Hat 7.3 and maybe into Red Hat 8.X some day. After I compile a driver, I know there is more to it than simply running **insmod** to have the driver module automatically load at boot time. Is there an example or procedure that walks one through what an install shell does? What config files are tweaked and where does the driver go?

—Steven Brown, sdbrown327@charter.net

Driver information goes in the /etc/modules.conf file. For example, the line **alias eth0 eepro100** specifies that the eepro100 driver should be loaded for use by Ethernet interface eth0. If the module is for your boot device, you need to use an initial ramdisk. See the mkinitrd man page.

—Robert Connoy, rconnoy@penguincomputing.com

Red Hat has assembled a guide to their boot script layout at www.redhat.com/support/resources/tips/Boot-Process-Tips/Boot-Process-Tips-3.html. You can insert your own commands into this file for execution at boot time.

—Chad Robinson, crobinson@rfgonline.com

This page, www.linux.org/docs/ldp/howto/mini/Modules, gives you some insight on the working of loadable modules. It has the basics. Another interesting page about Linux kernel modules is at www.luv.asn.au/overheads/kernelmodules.

—Felipe Barousse Boué, fbarousse@piensa.com

In Defense of Telnet?

I believe Mr Marti has been tempted into a somewhat facile response to the question regarding telnet [see Best of Technical Support, *LJ*, April 2003]. OpenSSH is indeed an excellent choice over rlogin and telnet outside a trusted environment. But inside such a network, rlogin, telnet and the other r* commands give excellent and much more convenient service within their respective realms of use. I would not be happy to retire rlogin and telnet especially where I have to deal with older systems other than Linux. As ever with security issues, the cost of protection (embodied, in this case, in the greater complexity and inconvenience of setting up the secure tool) must be weighed against the benefits accrued. Simply replacing rlogin and telnet with SSH is useless unless the network is comprehensively bolted down and firewalled. Again, if appropriate protection is to be obtained, a full analysis must be done.

—Bob Hepple, bhepple@freeshell.org

You have the option of inserting public keys, for user authentication, on the servers to which you connect. Or, you simply can use password authentication. Even if you use password authentication, the password is protected by symmetric encryption with SSH. Today's hardware adds an additional wrinkle to security. Are you sure your broadband router doesn't have a trojan installed to sniff telnet packets and pass them over the network? Are you sure your 802.11 wireless bridge isn't passing your telnet packet over the air? A lot of hardware

you have no control over has access to your data. Mr Hepple's assertion that “replacing rlogin and telnet with SSH is useless unless the network is comprehensively bolted down” is wrong. We always speak in terms of “raising the bar” of security and adding “layers of an onion”. You are correct, that if a security solution has high cost, one should perform a threat analysis, but SSH is so easy it is a no-brainer.

—Christopher Wingert, cwingert@qualcomm.com

Modern distributions include OpenSSH by default, and make you go through extra effort to run a telnet server. Thankfully, the convenient choice and the secure choice are the same. SSH with password authentication is as easy as telnet. With properly configured keys, it's easier. Check next month's *Linux Journal* for some productivity-boosting OpenSSH tips. If you need to log in to your Linux system from another OS, check Rick Moen's list at linuxmafia.com/pub/linux/security/ssh-clients for compatible software.

—Don Marti, info@linuxjournal.com

Archive Index Issue Table of Contents

Advanced search

New Products

Heather Mead

Issue #111, July 2003

Axis for Linux, Little Board 700, Mini-Box M-100 and more.

Axis for Linux

Axis Systems, Inc. announced that its Xcite, Xtreme and Xsim hardware acceleration and emulation systems now are available for Linux. Utilized by developers of complex electronic systems and system-on-a-chip designs, Xsim, Xcite and Xtreme allow designers to test and debug using a single system and database, thereby shortening the verification cycle. Xcite, Xtreme and Xsim together provide software simulation, accelerated simulation, system emulation and hardware/software co-verification for behavioral, RTL and gate-level designs with a runtime performance of up to 500k cycles per second.

Contact Axis Systems, Inc., 209 East Java Drive, Sunnyvale, California 94089, 408-588-2000, info@axissystems.com, www.axissystems.com.

Little Board 700

The Little Board 700 is a single-board computer (SBC) from Ampro Computers, Inc., designed for demanding, power-sensitive embedded applications. The Little Board 700 is an embedded SBC using the EBX form factor (5.75" × 8"). It is available with a variety of low-voltage processors: Pentium III with 512KB Cache at 933MHz, Celeron at 650MHz or Celeron at 400MHz. It supports up to 1GB DRAM and includes thermal monitoring and power management functions. An onboard Type II CompactFlash socket supports up to 1GB of Flash memory, accessed as an IDE hard disk drive. Onboard peripherals include dual 10/100BaseT Ethernet controllers, an AGP 4× video controller with flat-panel support, AC97 sound, two USB ports, four full-modem serial ports and a PC/104-Plus bus.

Contact Ampro Computers, Inc., 5215 Hellyer Avenue #110, San Jose, California 95138, 800-966-5200, info@ampro.com, www.ampro.com.

Mini-Box M-100

Mini-Box is an x86 computing platform designed for embedded or general-purpose computing applications. Mini-Box is based on a VIA Mini-ITX motherboard and runs at 12V, making it suitable for applications where small form factor, low power consumption and reduced noise levels are necessary. Additionally, Mini-Boxes are equipped with LCD displays and a customizable 14-key keypad that replaces keyboards or mice. Weighing two pounds each, standard Mini Boxes come with 256MB of PC133 RAM, 64MB of CompactFlash and an 800MHz x86 VIA C3 processor. Options include a 533MHz fanless processor, 128MB of CompactFlash and a 40GB 2.5“ IBM drive. Mini-Box runs a small embedded Linux OS or a full standard distribution.

Contact Ituner Networks Corp., 3071 Southwycke Terrace, Fremont, California 94536, 800-978-8637, www.mini-box.com.

SuSE Enterprise Server 8 for AMD64

In support of AMD's 64-bit Opteron processor, SuSE Linux Enterprise Server 8, based on UnitedLinux 1.0, now is available. Taking advantage of Opteron's capabilities to handle applications in both 32- and 64-bit environments, Server 8 is a complete server OS for Opteron processors. Server 8 comes with an optimized 2.4.19 kernel (including GCC 3.2.2) that supports high-availability work and high-performance interaction with storage systems by means of asynchronous I/O, multipathing memory access and the management of up to 600 physical hard disks. Server 8 provides scalability for up to 64 processors and up to 512GB of main memory.

Contact SuSE Inc., 318 Harrison Street, Suite 301, Oakland, California 94607, 888-875-4689, info@suse.com, www.suse.com.

HyperBlade Opteron Cluster, 1U and 2U Servers

APPRO announced the availability of its new HyperBlade Server Cluster, 1U and 2U dual servers based on AMD Opteron processors. The APPRO HyperBlade cluster solution is designed for the high-performance computational (HPC) market. The HyperBlade Server Cluster offers high-density architecture by using commodity x86 components in a single cluster. It supports up to 80 compute blades and up to 160 AMD Opteron processors. The APPRO 1U and 2U dual servers provide high-processor and memory bandwidth performance. They feature dual AMD Opteron processors, simultaneous 32-bit and 64-bit computing capability, HyperTransport technology, PCI-X support, swappable HDDs and up to 16GB of DDR SDRAM.

Contact APPRO, 446 South Abbott Avenue, Milpitas, California 95035, 800-927-5464, www.appro.com.

Mandrake Corporate Server 2.1 for Opteron

In tandem with the release of AMD's Opteron processor, MandrakeSoft, Inc. released Mandrake Linux Corporate Server 2.1. Server 2.1 is designed for high-performance applications supporting web servers, database deployment (MySQL-64) and application and file servers. It allows users to migrate to 64-bit technology by supporting legacy 32-bit applications as well as 64-bit ones. Based on the 2.4 kernel, Server 2.1 offers support for DHCP, Apache, Postfix and Squid proxy server; POP3, IMAP and web-mail services; and several journalized filesystems.

Contact MandrakeSoft, Inc., 2400 North Lincoln Avenue, Altadena, California 91001, 626-296-6290, www.mandrakesoft.com.

Aspen Systems Opteron Beowulf Clusters

In collaboration with AMD, Aspen Systems, Inc. announced an array of customized, Opteron-based Beowulf clusters and high-end server solutions. These clusters and servers feature 1U, 2U and 4U rackmount platform products with high-speed Myrinet interconnects; Aspen Beowulf Cluster Management software; AMD 64-bit technology for legacy 32-bit and new 64-bit applications; HyperTransport technology for increased communication speed between circuits; and an integrated 128-bit wide DDR DRAM controller, capable of supporting up to eight registered DDR DIMMs per processor.

Contact Aspen Systems, Inc., 3900 Youngfield Street, Wheat Ridge, Colorado 80033, 800-992-9242, www.aspsys.com.

Archive Index Issue Table of Contents

Advanced search