Advanced search

Linux Journal Issue #79/November 2000

Archive Index

Advanced search

Focus: Hardware

Don Marti

Issue #79, November 2000

If you're on a shopping trip for PC hardware, and you come across this magazine in your favorite computer store, read the hardware articles before you hit the aisles.

A man went to the doctor and said, “Doctor, it hurts when I do this.”

“Well, don't do that, then,” said the doctor.

Simple but good advice. In the world of Linux, many people subject themselves to pain and suffering trying to get Linux to work on crummy hardware, when they could just *not do that*, and instead do some research (*LJ* is a great resource) to find hardware that works well. That doesn't mean wizards who want to hack new drivers shouldn't try to get Linux working on bad, even pathologically stupid, hardware if they want. But if you're planning to get a new web server on the Net by Monday, don't start grabbing random crap off the shelf at Discount Computer Land on Sunday night.

If you're on a shopping trip for PC hardware, and you come across this magazine in your favorite computer store, read the hardware articles before you hit the aisles. Picking high-quality, Linux-compatible parts will save you a lot of time and effort, and will encourage the hardware vendors to test their stuff with Linux in the first place. Buy the magazine afterward, though. If you promise to buy the magazine and the store people say “Hey, this isn't a library,” you can point them to this:

Hello, nice computer store person. Please let this customer read the hardware articles because he or she promises to buy the magazine, and you might sell some hardware, too. Thank you.

The Duron is the “cheap version” of AMD's Athlon CPU. It's not as fast as the fastest Athlons or Pentium IIIs, it doesn't have as much cache, but it is very usable in a good basic desktop machine that runs StarOffice, the GIMP or your favorite development tools. ASL, Inc. has built a respectable Linux workstation around Duron, with top-quality parts and performance that's more than adequate for almost everyone.

Everyone, that is, except people who want the current top-of-the-line Linux machine. Jason Collins, Mike Higashi, Sam Ockman and I sat down for a fine dinner at Taqueria Los Charros in Mountain View to discuss hot hardware, cool cases and fans, and we got some good recommendations from Eric Raymond and Darryl Strauss, too. So check out the article if you're building a no-compromises workstation, and if you're ever in Mountain View, I recommend the super carnitas burrito.

Thinking of dual booting? Well, don't. If you're seriously into learning Linux, don't handicap yourself by putting it on a spare partition on the same machine with all your legacy stuff. The First Law of Dual Booting states that “The application you need is always on the other OS.” So where do you put Linux? On a cheap but stable Linux system, naturally. Jason Schumaker took a savage journey into the heart of the cheap hardware market, and emerged with “Return of Revenge of the Killer $800 Linux Box”--a good selection of high-quality parts that are trouble-free with any Linux software you care to name. That is more than you can say for most PC vendor's low-end desktop box du jour. Whether you're building a cheap box to learn Linux or building a main machine on a budget, Jason's selection is a good place to start.

Feature Articles on Hardware

—Don Marti

Building the Ultimate Linux Workstation

Don Marti

Issue #79, November 2000

We went to hardware experts to find the best and fastest components for building your own Linux workstation. Here's how to make a responsive system with high-performance 3-D, fast disk, lush sound and the little things to keep it trouble-free.

In 1996, Eric Raymond wrote a popular article for *LJ* called “Building the Perfect Box: How to Design Your Linux Workstation”. Lately, some of us have gone hardware shopping, and noticed that the hardware choices have changed a little since Eric recommended a Pentium 133 or 166 with 2GB of disk space. So, we asked Eric and some other experts to help us update the recommendations for building the ultimate Linux workstation.

CPUs

Back in 1996, Eric advised *Linux Journal* readers to “buy one or two levels lower than commercial state-of-the-art”. That's still good advice in general. If you're trying to save money, there's a lot of price difference between a manufacturer's fastest CPU and the ones just a little slower; and your money will be better spent on more RAM or other parts.

Even though Eric's advice is still good on the planet Earth, we're building the ultimate Linux box, so we'll select the fastest CPUs we can get. Yes, *CPUs*, plural. With background tasks on their own CPU, your cycle-hog processes, such as MP3 encoding and the GIMP, will have the other all to themselves, and that's good news. However, if you're on a budget and are going for a general feeling of “responsiveness” when starting programs or opening files, SMP won't help much, and you'll be better off with one CPU and 10,000RPM SCSI drives than with slow drives on an SMP system.

Which CPUs? Darryl Strauss worked on the render farm of Alpha-based Linux machines that rendered many scenes in *Titanic* (see the February 1998 issue of

Linux Journal). “I really love Alphas”, he says. “I have a dual 264 system at home and it rocks. They are also doing some great work on their upcoming boxes that will include faster processors, a faster bus and AGP graphics. The problem is that they are niche. If you have a specialized application that runs on the Alpha, you can't beat them. If you want a general-purpose system, then Intel-compatible is the way to go.”

Most of the people writing the software you want to run are testing it on Intel-architecture systems, so we'll stick with Intel's current top-of-the-line processors for now. AMD CPUs are binary-compatible with Intel, and performance-wise they can take Intel one-on-one, but currently there's no motherboard that supports two of them.

Motherboard

The list of available motherboards changes constantly. There are enough new motherboards coming out to sustain entire web sites with just motherboard news. How can you pick one? Jason Collins, a software engineer for VA Linux Systems and developer of the Cerberus test suite, is building his own Linux system for home use, and has some advice on selecting a motherboard. “I look at overclocking sites. If a board is stable when it's overclocked, it's going to be very stable at its intended speed”, he says.

Sam Ockman, CEO of Penguin Computing, says, “Spend some money, get a nice Tyan board.” Tyan motherboards have a reputation for reliability among the Linux system vendors. Mike Higashi is a Linux consultant who builds many of his own machines, and he recommends Asus boards.

Darryl Strauss is using the Asus P2B-DS, which is based on Intel's old but reliable 440BX chipset. The BX motherboards are still a decent choice for SMP on a budget. If you're not going SMP, the current CPU and motherboard combination of choice is an AMD Athlon on an ABit KA7-100.

When choosing a motherboard, be sure it supports ECC RAM, and enough of it for your foreseeable needs. Considering the amount of RAM in an ultimate Linux workstation and the length of time it runs, you will get bit flips if you don't use ECC. There's a new “Linux ECC” driver that provides extended logging of single-bit ECC errors, which are normally invisible. If you have a supported motherboard, you can use it for testing.

Microsoft Quality Hardware?

Video

Open up a large white window on your screen, and look over the top of your monitor at something in the distance. If you can detect flicker out of your peripheral vision, your video card or monitor is bad or misconfigured. Flicker that you don't ordinarily notice can give you a headache or other health problems. Different people seem to require different refresh rates—most are around 75-85Hz. The video card in any Linux box should be capable of an adequate refresh rate. For an ultimate Linux box, you'll need that high refresh rate on a very large monitor at 24 or 32 bits per pixel. Even if you aren't interested in 3-D, get a good video card to spare your eyes.

The ultimate Linux workstation needs 3-D. Darryl Strauss says, “The up-and-coming board to watch is the Radeon,” which is ATI's entry in the 3-D graphics performance race. “Performance is about the same as the GeForce2 and they want to do open-source drivers. It's just not out for Linux yet.” Watch your favorite Linux news sites for details. As we go to press, the speed leader is the NVIDIA GeForce2, which Strauss calls, “a great board except for the license”. This January, VA Linux Systems took a hit from the UNIX vendors' crack pipe and broke ranks with the rest of the Linux community to offer a fast but proprietary 3-D package for NVIDIA cards.

Trying to plug binary-only proprietary software into the low levels of a sleek open-source environment is like pilling a cat with chopsticks, so until NVIDIA gets on the clue bus, it's safest to go with one of the second-ranked boards—either the Voodoo 5 from 3dfx, or the new Matrox G450. Jason points out that the Matrox cards are good at taking advantage of more CPU power. “With a fast CPU, you can start to approach NVIDIA's speed.”

Audio

Creative Labs used to be in the same situation as NVIDIA is now. Their top-of-the-line product, the Soundblaster Live!, wasn't supported with open-source drivers. After realizing they have nothing to lose and much to gain, Creative launched the opensource.creative.com web site last year to host an open-source driver for the Live! There's now ALSA support, too. Creative is also cosponsoring a new standard called “OpenAL”, billed as “OpenGL for audio”, that will let developers create cross-platform three-dimensional sound applications. The Live! sounds great now, and will also be your ticket to the 3-D audio show of the future. Don't forget to write “Linux” on your warranty card.

If you just need a decent basic sound card, the days of scrounging antique ISA Soundblasters or messing around with ISA Plug-and-Play are over. The Creative Ensoniq AudioPCI, model ES1371, is an inexpensive, well-supported, and, as you might be able to tell by the name, PCI card.

SCSI or IDE?

Servers use SCSI, because it's the best way to connect a lot of drives. Cheap desktop machines (see Jason Schmaker's article, page 88) use IDE, because motherboards include an IDE controller for free and the drives are cheaper. But the place where people are actively debating SCSI vs. IDE is in midrange or high-end desktop systems.

The very fastest drives, at 10,000 RPM, are available in SCSI versions only. So, for a true ultimate system, SCSI is best. And as you add more drives to your system, you'll appreciate the fact that SCSI doesn't hog interrupts like IDE does.

When you compare mechanically identical SCSI and IDE drives, the IDE drive will have a faster raw transfer rate, because the interface is simpler. But Eric Raymond says, “As fast as disks are getting today, the difference is effectively noise. The real advantage of SCSI (and the reason it's faster overall) comes from its extra brains.”

For our ultimate Linux box, we'll choose the latest SCSI technology, Ultra 160. You don't need Ultra 160 for the amount of SCSI bus traffic created by two drives, but (1) this the ultimate Linux box; and (2) the price differences among SCSI cards are small enough that you might as well get a card that's going to last you through several sets of drives.

Rick Moen once advised me to select hardware that's flexible enough to be incorporated into the next Linux box you build. Some of the parts you buy for your ultimate workstation today could end up in your Freenet server next year. The Adaptec 29160 is a good example of this kind of (almost) future-proof hardware. When you do upgrade to 64-bit PCI, you'll still be able to use it. The 29160 is still new enough that the kernel you got with last year's Linux distribution might not recognize it, so you might need to upgrade. Check your distribution's hardware compatibility list to find out.

Symbios is a SCSI card vendor that's popular with the Linux hardware vendors. Their cards have good performance and are well-supported too.

The money you spend on a killer SCSI card is wasted without fast drives, so choose two or more 10,000RPM drives from IBM, Quantum or Seagate. One good rule of thumb is to choose the same drives the reputable Linux system vendors are putting in their servers.

If you do decide to save money and go IDE, stick to one of the top three drive vendors mentioned above. Earlier this year, kernel hacker Andre Hedrick, the maintainer of Linux's IDE driver, tracked a user's problem to the fact that Western Digital drives don't do error checking correctly. He posted to linux-kernel, “WDC drives blow off the CRC check of UDMA.... This is BAD and STUPID.” Western Digital fired back on their web site with, “If there's a problem using these drives in Linux the problem most likely lies with the software driver and not the hard drive itself.” I'm going to believe the kernel hacker over the hardware vendor, and stay away from Western Digital drives.

Eric's advice is to put your own precious files on one drive, and your distribution on the other. That makes restoring your system after a disk crash easier. “If you have two, maybe the crashed one was your system disk, in which case you can buy another and mess around with a new Linux installation knowing your personal files are safe. Or maybe it was your home disk; in that case, you can still run and do recovery stuff and basic net communications until you can buy another home disk and restore it from backups.”

And two drives are faster than one, if (as often happens) you're dealing with both /tmp and /home, or /home and /var, simultaneously. You can also use Linux's software RAID to mirror the same set of file systems on a pair of disks. Don't rely on a spare hard drive in the same machine as the only backup for the main drive, though. Often a power supply failure can fry both drives at the same time. Back up over the Net to another system, or use tape.

Network Card

The network card market is rather dull right now: Intel seems to dominate, despite some occasional complaints about hardware changing out from under the drivers and causing unpredictable problems with certain combinations of network traffic. If you're building your own box, you should be qualified to build a new kernel if you have to update the Ethernet driver.

Miscellaneous: Power Supply, Case, Cooling

Whatever you do, don't buy the latest hot CPU and drives and then mount them in a case with inadequate cooling, or hook them up to a flaky power supply that's going to blow out and fry everything. One of the ways a switching power supply can fail is to put the input voltage straight into the parts that are expecting 5 or 12 volts DC. That's not good.

If you live on the bleeding edge, hardware-wise, then your case, power supply and fans will outlast several motherboards and CPUs. Some people take the motherboard and CPU from what was a hot desktop box and use it in a web server. Again, think about being able to shuffle the components you buy today into another machine next year.

Mike chooses PC Power and Cooling power supplies, with Sparkle Power as a second choice. PC Power and Cooling has one called the “Silencer”, which makes a Linux box a little more pleasant to share a room with.

Choose a case with plenty of room to work inside, a side panel that's easy to remove, and several places to install fans. Mike builds quite a few heavy-duty Linux servers for small companies that don't have dedicated air-conditioned server rooms or rack space. That means he needs a tower case with a lot of cooling capacity, and he recommends Supermicro. They make a tower case with mounting points for fans in all the standard places, as well as blowing sideways across the drives. Direct airflow is absolutely required for the 10,000RPM drives that go into an ultimate Linux box. Otherwise, they will soon cook themselves.

A Canadian company, AMK, makes a series of cases called the “Overclocker's Dream”, with exhaust fans mounted in “blowholes” in the top of the case. If you don't mind keeping the top of your machine clear, blowholes are an efficient way to get the hottest air out.

Jason points out that adding fans can sometimes make heat problems worse. You can create a vortex that traps hot air near an important component or, worse, add too many exhaust fans and not enough intake fans. Hobbyists often add an extra exhaust fan or fans to try to suck out the hot air, but if you have too many fans blowing out and not enough blowing in, then the power supply, and maybe the CPU, can overheat. A typical ATX power supply has a fan on the back and a set of vents positioned somewhere above the CPUs. In normal operation, the power supply fan draws air from above the CPUs and exhausts it, cooling the power supply and helping to cool the CPUs. But if the system has too many exhaust fans, they create a negative pressure inside the case, cancel out the efforts of the power supply fan, stop the flow of air inside the power supply and turn it into an expensive toaster.

Two rules to keep in mind when adding fans to the case are first, balance the type and number of intake and exhaust fans; and second, remember that hot air rises—put intake fans low and exhaust fans high. And don't run with the case open any more than you have to. You're just opening it up to severe interference from other devices. “My cell phone rang and my computer went blaaargh”, says Jason.

Ultimate Linux boxes can be loud. Drives, CPU fans, front-panel fans, back panel fans...pretty soon it sounds like you're working inside a hovercraft. Jason Collins plans to install a “DigitalDoc 2”, available from AMK. It's a control panel that fits in a drive bay and provides thermostatic control for fans. His fans of choice are “badass” new ones from Delta that can move 32 cubic feet of air per second. Jason is also putting sound-deadening material on the sides of his case to keep it from resonating.

And, of course, you'll need a UPS. The most important benefit of having one on a workstation isn't to keep your system running during power failures. An extended blackout will drain its battery. The big benefit is that the UPS warns you when it's running on battery, giving you time to shut down cleanly.

Both APC and Tripp Lite make well-regarded UPSes with serial ports. Penguin Computing had to choose between them. Ockman says that his answer to “Why don't you sell our UPS?” was, “Why don't you GPL your control software?” So they did. APC won't even release the specs for their protocol. Until APC realizes that making their protocol available to open-source developers isn't going to result in the loss of their U-boat fleet in the Atlantic, let's put them with NVIDIA on the list of companies we'll be happy to support when they get a clue.

Resources

Don Marti is the technical editor for *Linux Journal*. He can be reached at info@linuxjournal.com.

AMD's Duron Processor

Don Marti

Issue #79, November 2000

A $70 CPU may be all that you need for responsive word processing, photo editing, and coding. We take a first look at AMD's Duron, an excellent choice for midrange desktop Linux machines.

I have seen the future of desktop Linux systems, and its name is Duron. A lot of economy-minded Linux hobbyists became AMD K6 fans back when Intel's offerings were either pricey Pentium II or wimpy early Celeron. AMD is still cranking out perfectly respectable K6-2s and -3s that will do very nicely for most of today's applications. But, software gets bigger about as fast as users get less patient, so the Duron looks ready to step into K6's role as the CPU of choice for people who need responsiveness more than extreme processor power. That is, most users. A CPU that's too fast is a waste of money, so you're better off sticking with the economy CPU and spending your money on the other components.

A 600MHz Duron costs somewhere around $60-$70 (US) right now, and this is cheaper than the Intel Celeron. And clock-for-clock, Duron beats Celeron. So if you need an “Intel Inside” sticker for some reason, buy that separately instead of paying the extra 20 bucks to get the CPU that comes with it. Basically, a Duron is an AMD Athlon with a couple of minuses and one plus. It has less cache than the higher-priced Athlon and is not available at the fastest clock speeds. But, Duron comes in the “Socket A” form factor, which just seems sturdier than slot CPUs and is lower to the board, allowing for smaller cases.

All this would be very nice for hackers, but not terribly relevant for the business desktop market, if all we had was last year's desktop applications. But now, Sun's decision to release StarOffice under the GNU GPL is a big opportunity for IT departments looking to save money at the cost of subjecting Microsoft-Office desktop users to a little pain and suffering in terms of compatibility and training.

Most corporations are probably going to play it “safe” for now, and throw money at licensing and Windows support staff to preserve the Microsoft Office status quo. If your company has an Excel or Access power user among the executives, that's all you can do, short-term. Just as companies moved out of vibrant cities into brain-dead suburbs in the 1970s to be close to where the boss lives, your employer will probably keep Microsoft Office because that's what the executives know. But, if your big cheeses don't have any computer skills, they're going to complain equally about whatever desktop applications you give them. You might as well go with the no-license-fee, easy-to-support GPL alternative, so they don't complain about the IT budget too. Be realistic.

StarOffice and Duron are the enabling technology for a big boom in Linux desktop systems. But Linux vendors are all building servers, servers, servers, right? Wrong. ASL, Inc. is a Linux vendor that has been quietly plugging away since 1995. Even though most of the other Linux vendors have been racking up server sales almost exclusively, ASL's President and CEO, Jeff Nguyen, says that their business has been about 50/50 servers and workstations. The company recently chose the desktop-friendly Mandrake distribution as its default install. They're now rolling out a line of Duron-based desktop systems.

ASL's Duron box came with a slick, professional out-of-box experience. The packing slip, complete with a full list of components, is a good thing to have on file in case you ever need to reinstall software. The mouse, keyboard, power cord and manuals were packed nicely on top of a convenient lift-off cardboard insert, and the machine itself came out and practically set itself up.

I can never resist taking a peek inside the case when the system arrives, so one thumbscrew and two Philips later, I was looking at the ASL system's guts. The case opens up all the way around—pop the thumbscrew and the top slides off, then the sides slide up. Quality components throughout—IBM hard drive, Matrox video card, Ensoniq sound and the same Intel Ethernet card everybody uses. Motherboard chipset by VIA, a company that built a good reputation for solid K6 chipsets. If you're like most Linux users, you never put all the screws back in, so rest assured that this case stays together with just the thumbscrew.

But is it fast? Realizing the essential bogusness of benchmarks as applied to regular desktop use, I ran the GIMP, xaos, and other CPU-hog programs on the

Duron, alongside the same programs on a much more expensive Athlon system clocked at 800MHz. And you know what? The speed is indistinguishable. If you want to crack RSA keys or do heavy rendering, you should probably look at times posted by people who use your software of choice; for regular use, just get the Duron. The pre-GPL StarOffice 5.1, in particular, was nice and responsive, and I would be happy to find this machine in my cubicle upon starting a new job.

Although it has one little Duron CPU and one IDE hard drive, the fans installed on the prototype look adequate for a much hotter system. It's a big case, about the same size as the one VA Linux Systems used for dual Xeon workstations. And the fans, front and back, are loud. Like web server loud. All this to cool one IDE drive and one little socketed CPU?

Not for long, Nguyen says. ASL's future Duron machines will use a Micro ATX version of the same motherboard, so ASL will be able to make almost identical Duron-based systems in new, smaller cases with ElanVital's "Whisper Technology". ElanVital claims that they can reduce overall noise from 37 decibels to about 30 decibels with the Whisper system, which consists of a CPU heatsink and fan, a flexible tube for bringing outside air to the CPU fan, and a power supply with a thermostatically controlled fan. Good to see somebody's thinking about the human factors. Any Linux box people have to work next to should be quiet. I said, any Linux box people have to work next to should be quiet.

The noisy fans are going away, but I ran into a couple of problems with ASL's preinstalled software. First, the kernel wasn't configured to support DHCP, which is surprising for a desktop system. However, you can expect DHCP out of the box if you order an ASL system now; John Kim, a Linux system engineer at ASL, says they'll be making DHCP work out of the box now that I've brought it to their attention. An easy fix for them to do, since anybody can customize Linux.

Second, I ran into a problem with mangled fonts in StarOffice when running the Accelerated-X from Xi Graphics that ASL installs by default. I reloaded the box with the stock Mandrake disks supplied, and the problem didn't show up under XFree86. Kim says that ASL is going to try to replicate the problem and check into it with Xi Graphics, but no word back from them yet.

Fortunately, this is not really a problem. You can order the system without the proprietary X server, and if you're going out on a limb to put free software on the business desktop, why leave yourself with one piece of proprietary software to track? The Accelerated-X install requires you to enter a serial number from the package, which is just not worth the hassle. XFree86 works, works well and installs automatically if you set things up right. Even if you're not going to hack or customize it at all, proprietary software bought and paid for is still inferior to the free stuff because of the administration burden of keeping track of licenses.

Despite the glitches, I can recommend this machine for almost everyone, and especially for people who have to keep track of an office full of them. Duron is a good CPU, all the other components are top-quality, and ASL has experience in the Linux desktop business. Order one of ASL's Duron boxes with your company's choice of free software—they'll do custom loads for you—but be sure to ask them to hold the proprietary X server. Otherwise, what's the point?

Don Marti is the technical editor for *Linux Journal*. He can be reached at info@linuxjournal.com.

The Return of the Revenge of the Killer $800 Linux Box

Jason Schumaker

Issue #79, November 2000

Can't afford Don Marti's “Ultimate” Linux box? Well, read on—Jasonw outlines options for the economically challenged.

Don Marti's a big spender. In his article, “Building the Ultimate Linux Workstation”, he waxes technical, recommending various components needed to build the mother of all Linux workstations (see page 80). He has little regard for expense, and his Linux box is top-of-the-line all the way. That's all well and good, but the thing would cost some unthrifty soul at least $3,000 (US). If you have that kind of cash, Don can tell you where to spend it, so read his article (before reading this), as he provides good general advice on selecting components for your system.

Of course, reading Don's article will no doubt set your mouth to watering, but the hefty price tag may send you back my way. I'm bitter, but resigned to the fact that I simply cannot afford Don's screamer of a system. However, I needed a computer for home, and didn't have much money. To be exact, I needed to stay under $1,000. So I began searching for components and discovered that putting together a zippy, dependable Linux box can be done for as little as $800. I checked out workstations from VA Linux and Dell, amongst others, but found that by putting it together myself, I could save $500 for a comparable system. More importantly, I'm building the system from scratch. I chose the components and I'm responsible for putting them together. It wasn't terribly difficult and the knowledge gained is immeasurable. How much do I love my new computer? Very, very much!

Options for Penny-Pinchers

Sacrifices are in order when building an economical system. To save money, you shop for what you need, not for what you want. This means looking for discontinued parts or “last year's models”. It also means locking up your ego.

Sure, I would like to have two CPUs and multiple SCSI hard drives, but I don't *need* that stuff.

I'm not into gaming and I don't do much fooling around with the GIMP. I write, I surf, I e-mail. This means one CPU, one hard drive, decent video and sound cards, a solid motherboard, a refurbished monitor and as much memory as can be afforded. Yes, refurbished. Not ideal, but you can spend as little as $150 for a sufficient monitor. However, we are talking about your eyes, so take care to spend as much as you can—make the monitor a top priority. I was lucky enough to have a spare monitor, but I did some searching and good deals abound.

Research is very important. Take some time to find the best deals. At one point, I had found a processor for $69 and nearly bought it. The next day I stumbled upon the same processor for $56! If you really need to save money, exhaust your resources. But you should be able to stick with companies you recognize and trust.

I spent a week scouring the Internet, and bought my components from various sites. My components arrived within a week of placing the order. Sure, I paid for shipping, but I still saved money, since Seattle computer stores just weren't price competitive. The Internet connects you directly to hundreds of stores with sales, discontinued items and so on. Working with a low budget put me at the mercy of Internet shopping, but I have no complaints thus far. If shopping via the Internet scares you, then research the companies you buy from—it will only take a few minutes. The system I put together consists of the following components:

Motherboard

Type: Super Socket 7 503+ Baby AT**Price:** $80 US (http://www.aberdeeninc.com/) **Specs:** VIA Apollo MVP3 chip set, 2-DIMM and 4-SIMM sockets, 3x PCI, 3x ISA, 1x AGP, ECC and PC100 memory support, and up to four IDE drives

Linux Journal has used Socket 7 motherboards for years. They are workhorses and the one I chose is both affordable and powerful. This is a step or two down from the top, but the board is solid. Its main strength is memory. There are four (72-Pin) SIMM sockets and two (168-Pin) DIMM sockets, which allows for as much as 512MB of system memory. Cache memory is 1MB. This all means my initiation into the world of Quake may not be too far off. It also means that this board should allow me to run most of the applications I want. The board will accommodate up to four IDE drives, and works with AMD K6 processors.

Processor

Type: K6-2 3-D 500MHz**Price:** $56 US (http://computersupersale.com/) **Specs:** 321-Pin (CPGA)/ZIF Socket 7 (P54C/P54CS/P55C MMX), RISC86, 64-bit (pipelined) 100MHz, On-Chip Split 64KB (L1) Cache

Don Marti says, “A CPU that's too fast is a waste of money, so you're better off sticking with the economy CPU and spending your money on the other components.” So, I bought an AMD K6-2 processor. It works well with the Socket 7 motherboard and was a mere $56! I do not believe the chip is discontinued, but it doesn't receive the same marketing push as, say, the Duron. Less marketing, in this case, equals lower cost. It doesn't mean lower quality.

I wasn't too worried about having a superfast processor, since my needs don't require it, but this processor should run Linux fast enough and, more importantly, the low price allowed me to spend more on memory.

Memory

Amount: 128MB**Price:** $140 US (http://solutions4sure.com/) **Specs:** 100MHz DIMM, 1 module, 100MHz memory bandwidth

Get as much memory as you can afford. I planned my spending in order to afford at least 128MB, which allows me to run multiple applications simultaneously. For $140 I have as much memory as I should need, for now. However, my motherboard supports up to 512MB, so once my raise goes through, I should be able to invest in more, more, more.

Hard Drive

Type: EIDE**Price:** $80 US (http://www.egghead.com/) **Specs:** Ultra ATA/66 interface, 15.3GB, 512KB cache buffer, 5,400RPM

Ideally, I would like to have two drives, but this is not the “ultimate” Linux box, so one will have to do. Money constraints pushed me toward IDE, instead of SCSI, but at least I was able to get 15.3GB for a low price. Another drive can always be added later (buy a case with space), which would allow me to run Linux on one and store personal files on the other. As Don Marti recommends, “Don't rely on a spare hard drive in the same machine as the only backup for the main drive...backup over the Net to another system, or use tape.”

Sound Card

Type: Ensoniq AudioPCI**Price:** $24.99 US (http://www.computersupersale.com/) **Specs:** 2MB or 4MB memory, 16-bit stereo digital audio, MT-32 compatible instrument set

This was a no-brainer, since Creative is one of the better sound card manufacturers. Don Marti rates the Soundblaster Live! as the card for his ultimate box and I have jumped a few notches down with the AudioPCI card, which he recommends. This is a good, basic card that serves up CD-quality sounds and getting it to work with Linux is easy. It's perfect for a low-end Linux box and should meet most user needs. Again, the price is right—just $25! Don't forget to write “Linux” on your warranty card.

Video Card

Type: ATI 3-D Rage IIC**Price:** $45 US (http://www.aberdeeninc.com/) **Specs:**8MB, 66MHz AGP Bus, 528MB/sec. peak, MPEG/DVD acceleration

The video card is the only part of the machine that I am not completely satisfied with. I went with the ATI card to save money, but it just isn't the right one, I fear. The verdict is still out, but scrolling up and down, as well as cursor movement is slow. I don't know how much patience I will have for that, so I will be changing the card as soon as I can afford one. I most certainly am not against all ATI cards, so their high-end cards may be worth a look.

Dan Wilder, *LJ's* technical manager, recommends offerings from 3dfx or Matrox. These cards will be more than double the price of my ATI card, but well worth it for gamers. As with a monitor, the video card will have a direct relationship with your eyes, so I recommend spending a bit more on graphics. A lot more if you're playing games. The card I chose will work fine for my needs, but a 16MB card would be better for gaming.

Case and Fans

The only mistake I made while ordering components was committed on the case. First, I should have bought one locally, which would have saved shipping costs. Duh! Second, I bought an ATX case, which won't work with my AT motherboard and added to my shipping costs. Doh! I returned the ATX case and cruised over to a local shop for a mid-tower AT case. It set me back $30 and included a 300W power supply. It has a removable side panel, room for extra drives, and space for multiple fans. It isn't see through or blue, nor does it resemble a penguin. Functionality before style is the motto of the thrifty spender.

Don Marti (page 80) has useful recommendations for the number of fans your system will need. I currently have three (one for the CPU, one mounted toward the front and one nearer the back) and am considering adding another. They are cheap and keeping your system cool is too important. All in all, I spent $50 on three fans.

The Rest

I *do* need a modem, but haven't bought one yet, which means I am without the Internet at home. This explains why I'm at work on a Sunday, as well. I intend to buy a US Robotics v.90 external for about $75. Using an external modem has the benefit of reducing the temperature within your case and provides easy access for you. Plus, most external modems are compatible with Linux and all external modems with a serial port are compatible with any device with an RS-232-compatible serial port (i.e., the Palm PDA).

I already had a keyboard and mouse, but these are cheap and would hardly add much expense. The same can be said for a floppy drive, which I found new for $10. You will probably want a CD-ROM, especially for installing your distribution. I had an older one, which worked fine for installation purposes. However, if you can get by without a CD-ROM, you might be better off saving up for an external CD burner, which is my plan.

The Results

While Don was out spending thousands of dollars on his system, I spent hours and hours searching for the lowest-priced components, in order to keep my system low-priced. All told, this isn't a screamer of a system, but at least it yells. For under $1,000, I managed to put together a respectable system that I intend to show off to friends and family. It's mine—I built it!

Resources

Jason Schumaker (info@linuxjournal.com) is assistant editor at *Linux Journal*. He recently bought a 1965 Ford Galaxy 500 and has begun the restoration process. He is currently looking for right exhaust manifold. Anyone?

2000 Readers' Choice Awards

Heather Mead

Issue #79, November 2000

Enough about us already; what do you think?

Roll out the red carpet, it's time to announce the winners of the 2000 *Linux Journal* Readers' Choice Awards. After another explosive year, the hype and trendiness have cleared to reveal (as we knew it would) that Linux is a serious contender in almost every market. Although some speculate that the development of all things Linux has settled down, we know the revolution is gaining breadth and speed. While we all look forward to the new applications, devices and services on the horizon, these awards are an opportunity to appreciate what we already have. Judging by this year's list of choices (the longest ever), not to mention all those write-ins, we have a lot. To get an idea of what voters said, I've included some quotes for each category.

Over 4,000 readers voted in 24 categories on everything from favorite programming language to favorite game. What overall moral can we discern from this year's responses? The percentage of voters with fervent opinions is directly proportional to those with caffeine addictions. Coincidence...I don't think so.

Favorite Distribution: Red Hat Linux

> “I love them all.”

Red Hat Linux regains the top spot, the first time since 1997, with over twice the votes of the second-place distribution, SuSE. Mandrake doubles its percentage from last year to just over 14%, placing it third. Last year's winner, Debian GNU/Linux, falls to fourth place. Slackware has a respectable showing with 8.5% of the total vote. The most popular write-ins are FreeBSD and “roll my own”.

Favorite Office Suite: StarOffice

> “StarOffice but quickly becoming KOffice.”

With over 63% of the votes, StarOffice is the clear winner; WordPerfect comes in second with only 12%. The office suite is a sore spot for many Linux users, as shown by the write-in comments. Quite a few voters apologize before picking Microsoft Office as their favorite, and almost as many decry, “None, they all suck.” Emacs, vi and clones, and GNOME Office Suite all make write-in appearances.

Favorite Desktop Environment: KDE

> “Command line.”

While some proclaimed KDE dead with the announcement of the GNOME Foundation this past August, *LJ* readers chose KDE as their favorite desktop environment for the third year in a row. With 400 fewer votes, GNOME takes second place. Window Maker and Enlightenment are only 40 votes apart, but neither received more than 9.7%.

Favorite Word Processor: StarOffice

> “WP8, only not that Windows junk.”

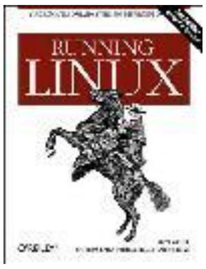
StarOffice claims a resounding victory in the word processor voting, racking up almost twice as many votes as its nearest competitor, WordPerfect. After those two, however, the results split into many processors with a few votes each, suggesting that this is one of the more personal categories, where everyone has an old favorite.

Favorite Text Editor: vi and clones

> “Elvis—so good it deserves its own category!”

If people are loyal to their old favorites in any category, it's got to be this one. **vi** and clones, a broad list, is once again your favorite text editor, with almost 40% of the total votes. Emacs are popular in all their forms, Gnu Emacs, X Emacs and the LaTeX+Emacs combo. Some write-ins express a desire to see some vi clones, like Elvis and VIM, given their own, separate listings. Maybe next year.

Most Indispensable Linux Book: Running Linux

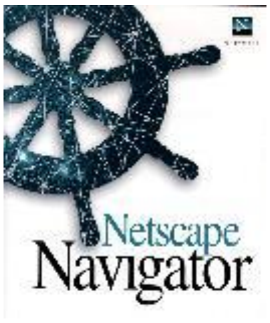


> "What are these book things you refer to?"

> "The whole O'Reilly series—life without them would be a disaster!"

Running Linux, by Matt Welsh, takes first place and *Linux in a Nutshell*, by Ellen Siever et al, takes second, in a reversal of last year's top two books. The margin was close though, with only 34 votes separating them. Looking at the complete list of vote-getters, the most extensive list for any category, many books appear indispensable to *LJ* readers. But not everyone is an avid Linux book reader; numerous write-ins show up for man pages and on-line documentation.

Favorite Web Browser: Netscape/Mozilla



> "Zen"

> "It's more of a necessity kind of thing."

Judging by the write-ins, this category resembles another ballot choice U.S. voters face in November: the lesser of two evils. Netscape/Mozilla wins by a mile (over 80%), with the next closest browser, Lynx, receiving 201 votes. Konqueror racks up 77 write-ins, just cracking the top five. The rest of the write-ins are almost evenly split between messages like "Mozilla—NOT Netscape", "I hate'm all" and "Internet Explorer—Sorry!".

Favorite *Linux Journal* Column: Kernel Korner

> “The nice thing is that with every issue another column stands out.”

> “I must say, without being too forward, I enjoy everything you all do.”

Aw shucks...you're making us blush with all this praise. Now, we know you're not always happy with every issue (a certain cover shot seemed particularly upsetting to some), but you seem satisfied overall. Kernel Korner receives the most votes, making it the favorite *LJ* column four years in a row. Second and third place go to At the Forge and Best of Technical Support, respectively.

Favorite Distributed File-Sharing System: Gnapster

> “What are you talking about?”

The Napster lawsuit gave rise to wide-spread controversy; it seemed like everyone had an opinion about it this summer. (You know something's up when Courtney Love sounds almost rational.) In our on-line vote, Gnapster proves to be the most popular file-sharing system with 45% of the votes. Gnutella comes in second with 34%.

Favorite Programming Beverage: Coffee

> “Sadly enough, canned capitalism (Coke).”

> “Sprite mixed with Fun Dip (cherry flavor).”

> “Mountain Dew! Not just another soft drink.”

Perhaps some of the cranky write-in comments can be traced back to the results of this category. Stereotype be damned, we like our caffeine and if it's got sugar, even better. Coffee is the beverage of choice with almost 50% of the votes. Other soft drinks come in at a collective number two, although many write-ins do not like to see their beloved Mountain Dew collapsed into this larger category. Water makes a surprising appearance in the number three spot.

Favorite Linux Game: Quake 3

> *“No time for games.”* (But plenty for questionnaires?)

> “Install the operating system.”

In 1998, it was Quake and in 1999, Quake 2; this year, of course, the winner is Quake 3. X-Bill pulls 9% of the votes and Civilization: Call to Power takes 6%, to place second and third, respectively. Rounding out the top five are the free games NetHack and FreeCiv. We received many write-ins, suggesting, happily, that Linux gamers have more reasons than ever not to leave the house.

Favorite Linux Web Site: Slashdot.org

> “eLinux.com. Talk about getting all the buzzwords in your domain name.”

> “Are you kidding? All of them!”

While not exclusively about Linux, Slashdot is where we all go, judging by the vote tally. Slashdot easily claims victory for the third year in a row, receiving twice the number of votes as the second-place site, Freshmeat.net. Other popular sites include LinuxToday.com, the Linux Documentation Project and Linux.org. Favorite web site is another category with an extensive write-in list, of which the most mentioned is linuxfr.org. *Tu parles français, non*?

Favorite Instant Messaging Client: Xchat

> “What else but Gnomeicu?”

A new category to the Readers' Choice Awards, just under half of all voters picked a favorite IM client. Xchat garners the most votes, picking up 20% of the total. Chomping on Xchat's heels, however, is Jabber; a mere 15 votes separate them. BitchX comes in third with 14. As for write-ins, gnomeicu is the most popular. Will everybody have a favorite next year?

Favorite Programming Language: C/C++

> "Plain C (without the ++)."

The perennial C/C++ wins 40% of your votes this year. To everyone who took the time to remind us that C and C++ are not the same language, we hear you loud and clear. Second and third place go to Perl and Java, while Python continues to expand its fan base by claiming 8%.

Favorite Platform: Intel x86

> "Amiga :) Running RH 5 and NetBSD 4.x."

> "Platforms that are really tall."

While some of you claim to use it under protest, Intel x86 is the clear winner with 60% of your votes. The clone AMD, a popular write-in last year, made it to the official ballot this year and received just under 20%. Rounding out the top picks are PowerPC and Alpha. As a write-in, Transmeta makes its debut in the platform category—a preview of next year, perhaps?

Favorite Shell: Bash

> "Turtle Shell."

> "Pathetic, Hollow Shell (of a Man)."

Bash won? It wasn't even close? You don't say. For the third year, the Bourne Again Shell demonstrates its hold over voters (and users), claiming 78% of your responses. In the distance are tcsh with 10% and ksh with 4.5%. A few write-ins express the preference to use bash for scripting while using another shell for everyday use. As for all the shell pun write-ins, who says geeks can't make a (bad) joke?

Favorite Development Tool: Gnu Compiler Collection (GCC)

> "VIM rules forever!"

Development tool is one of several categories this year where the distance between first and second place is more than substantial. GCC, at 71%, received almost ten times the number of votes as the second-place tool, Code Warrior. And really, where would any of us be without GCC? An up-and-comer, KDevelop, grabbed third place through the power of write-ins. We'll add it the list next year, guys.

Favorite Audio Tool: XMMS

> “My plain old radio.”

> “One that works would be nice!”

Things sure can change in a year. In the 1999 Awards, XMMS appeared a few times as a write-in; this year, it's the clear and away winner. Receiving just under 50% of your votes, it wins by a healthy margin, too. In the closest finish of any category, second and third place were determined by a single vote—Real Audio and mpg123, respectively. And some of you are still gladly using your CD players.

Favorite Graphics Tool: GIMP

> “My girlfriend says GIMP because she uses it (and is looking over my shoulder).”

> “Photoshop...sob sob. Why won't they port this thing?!”

So this new graphics tool came out of nowhere...maybe next year. For now, the GIMP has a strong-hold on this category; this year it wins with 72% of the total votes. The next closest favorite tools are xv with 10% and CorelDRAW with 7%. Voters are pleased by the GIMP's versatility and its ease-of-use, especially for the less artistically gifted among us.

Favorite Database: MySQL

> “The one that I don't need to admin.”

> “FBI Database on Un-American Activities.”

Some database users engage in a war of words when it comes to MySQL vs. PostgreSQL. Our readers, at least the ones that voted, prefer MySQL by a 2 to 1 ratio. Oracle 8i R2 comes in third with almost 11%, making our top-three line-up a repeat of 1999's. None of the other databases received more than 3% of the tally.

Favorite Ad Filtering Tool: Junkbuster

> “I don't use any (yet). Thanks for the tip.”

Judging by the low turnout for this new category, it seems like most of you use the old standard, manual filtering method: closing your eyes. Those of you who did express a preference choose Junkbuster as your favorite, beating SquidGuard by 40%. The most popular write-in filtering tool is AdZapper. Quite a few indies take the DIY approach and write their own proxies.

Favorite Communications Board: Cyclades

> “Federal Communications Board (or is that Commission?).”

Um...that's not quite what we meant. To be fair, though, some of you wrote that you didn't know what we meant by this category, which could explain why less than half of all voters picked a favorite board. Among those who figured us out, Cyclades is the number-one choice, with 50% of your votes. Digi International comes in second and Boca rounds out the top three.

Favorite Backup Utility: tar

> “Backups?!? Vee don't need no stinking backups!”

Okay, okay. While “real men don't need backups,” some of our more cautious readers like **tar**. This easy solution was quite vocal in 1999's write-in votes, so we added it to the official list this year—and it won. **tar** displaces last year's winner, BRU, by 46%, but BRU still comes in second. Arkeia and Amanda competed for third place, but Arkeia claims it with 11 more votes.

Favorite X-Server: XFree86

> “Only use XFree86...maybe others are good; dunno.”

Do any of you guys use anything but XFree86? Apparently not. 93% of respondents choose XFree86 as their favorite X-Server. Accelerated X, in second place, acquires votes totaling 3.5%. The most popular write-in—it only took six votes to get there—is Xpmac.

More information about the Readers' Choice favorites and other Linux-related products and programs is available on our Linux Resources web site, http://www.linuxresources.com/

Moving to Seattle to study American literature, ***Heather Mead*** *decided working was more fun than graduate school, and somehow became an associate editor at Linux Journal*. She's working on a screenplay, but who isn't?

Penguin Playoffs Return

Doc Searls

Issue #79, November 2000

Athletic flightless waterfowl? No, just the skinny on the Comdex exhibitor awards ceremony.

Properly, Penguin Playoffs should be held in Antarctica with real penguins, but we figure there isn't much box office in that, especially since these well-dressed birds are milled so precisely to form that telling winners from losers is impossible unless some are dead. In that case, the whole deal isn't much fun.

So we hold our Penguin Playoffs in Las Vegas, where the climate hasn't been hospitable to coldwater birds since the mid-Pleistocene. Not that this matters, since only penguins in the shape of Linux products are eligible to participate.

Actually, they already held the competitive part of the playoffs, and this year's winners have been selected. The playoff event will be an award ceremony and dinner during Linux Business Expo, which coincides with the November Comdex show.

To recap last year's inaugural playoff, Linus was there to hand out the awards, which coincided with *Linux Journal's* Editors' Choice awards (seemed like a good idea to honor two birds with one event). The winners were:

- Best Web Solution: TurboCluster Server from TurboLinux
- Best Hardware Solution: The Happy Hacking Keyboard
- Best Office Solution: Appgen Business and Accounting Applications
- Best Overall Solution: Global Media Corporation

This year we've doubled the number of categories, more or less keeping pace with the increasing diversity of Linux solutions now competing in the penguin businessphere. Here they are:

- Best Web Solution

- Best E-Commerce Solution
- Best New Product Announced at the Show
- Best Server
- Best Workstation
- Best Business Application Development Tool
- Best Embedded Solution

Why change categories in addition to doubling them? Well, markets are conversations, and these new categories seemed to better label the leading topical categories we're talking about today. No doubt they'll be more numerous next year and equally unlike this year's categories.

The awards event will be held at Gameworks on the strip again this year. Be sure to check our web site, http://www.linuxjournal.com/, for further details about day and time. Again, we will be giving out our Editor's Choice awards during the same event. There will be a *Linux Journal* party after the ceremony. Stop by the *Linux Journal* booth to find out when and where that will be.

We look forward to seeing you there.

Doc Searls (info@linuxjournal.com) is senior editor of *Linux Journal* and coauthor of *The Cluetrain Manifesto*. His opinions are his alone and do not express those of *Linux Journal* or SSC.

LTOOLS - Accessing Your Linux Files from Windows 9x and Windows NT

Dr. Werner Zimmermann

Issue #79, November 2000

If you work with multiple platforms, LTOOLS may offer a way to make your life a whole lot easier.

The **LTOOLS** under Windows provide a functionality similar to the **MTOOLS** under Linux: They let you access your files on the “hostile” file system.

Using LTOOLS from the Command Line

At the heart of the LTOOLS is a set of command-line programs, which can be called from DOS or from a DOS-Window in Windows 9x or Windows NT. They provide the same functionality as the well-known LINUX commands **ls**, **cp**, **rm**, **chmod**, **chown** and **ln**. Thus, under DOS/Windows you can do the following:

- list Linux files and directories (command: **ldir**)
- copy files from Linux to Windows and vice versa (commands: **lread**, **lwrite**)
- delete or rename Linux files (commands: **ldel**, **lren**)
- create symbolic links (command: **lln**)
- create new Linux directories (command: **lmkdir**)
- modify a Linux file's access rights and owner (command: **lchange**)
- change the Linux default directory (command: **lcd**)
- set the Linux default drive (command: **ldrive**) and
- show your hard disk partition setup (command: **ldir -part**)

As with many UNIX tools, these functions are included in a single executable, which is called with a bundle of command-line parameters. To make life easier, a set of batch files (shell scripts) are provided so that you don't need to remember and type in all these parameters.

Additionally, there is a UNIX/Linux version of the LTOOLS, to be used under Solaris or even Linux, when you want to access a file on another hard disk partition without mounting this partition.

LTOOLgui—A Java GUI for the LTOOLS

Some may feel that command-line programs are old-fashioned and ask, “Where is the LTOOLS graphical user interface?” Well, no problem: Use **LTOOLgui**. LTOOLgui, written in Java using JDK 2's Swing library, provides a Windows Explorer-like user interface (Figure 1). In two sub-windows, LTOOLgui shows your DOS/Windows and your Linux directory trees. Navigating can be done with the usual point-and-click actions. Copying files from Windows to Linux or vice versa can be done by copy-and-paste or by drag-and- drop. Clicking the right mouse button will open a dialog to view and modify file attributes like access rights, GID or UID. Double-clicking on a file will start it, if it is a Windows executable, or open it with its associated application. This even works with Linux files if they have a registered Windows application.

By the way, you can also use LTOOLgui as a file manager under Linux. As the LTOOLS command-line programs also come in a Linux version, you may access files on disks without mounting them.

I chose Java for LTOOLgui, because Java is especially suited for low-level hard disk access...only joking! No, of course, this is not possible in Java at all. If you want to access hardware directly, you have to use C++ code and JNI (Java to Native Interface). However, as the JNI only works for 32bit code, under Windows 9x this would mean using 32-bit to 16-bit thunking (see below). As I did not like the idea of combining Sun's Java with Microsoft's MASM code, I took another approach. I simply used LTOOLS' command line program, which gets called from Java via the well-known stdin/stdout- interface. So for the Java side, hardware access means simple stream-based file I/O.

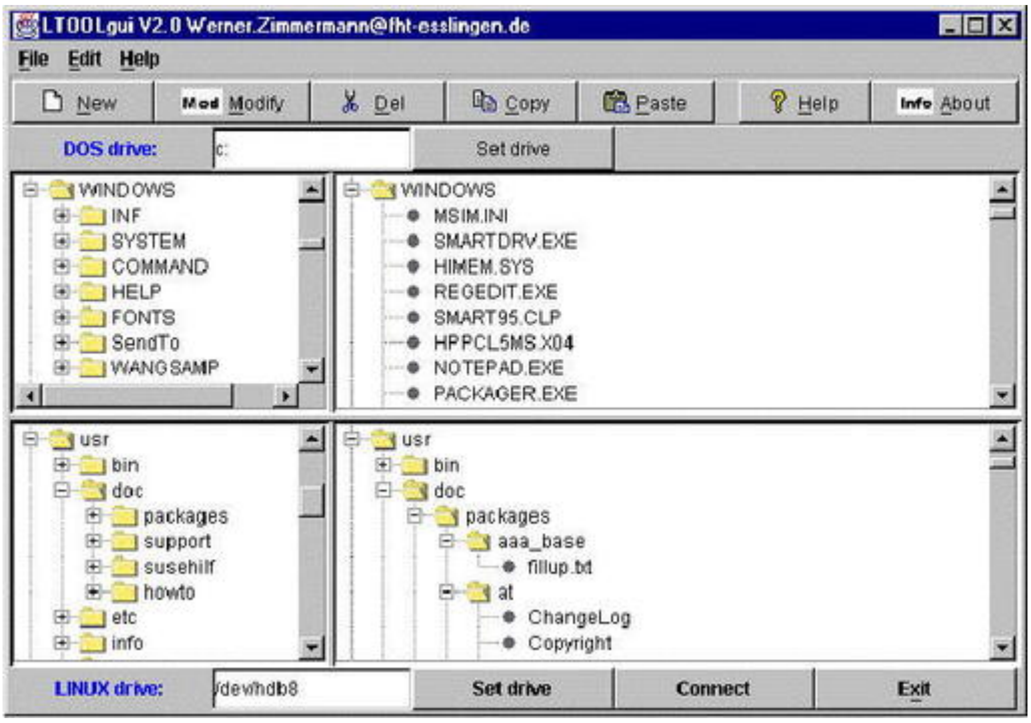


Figure 1. Java-based LTOOLgui Graphical User Interface

File Access over the Internet?

No doubt, any state-of-the-art program must be Internet aware. Well, if you run **LREADjav** on a remote computer and connect to it via LTOOLgui's connect button, you may access Linux files on this remote server as if they were local. LREADjav is a simple server dæmon, which translates requests, issued by LTOOLgui over TCP/IP, into LTOOLS command line program calls. The output of the command line programs is sent back via TCP/IP to LTOOLgui (see Figure 2). Of course, you can not only view directory listings, but also do everything remotely, that you can do locally, including file upload and download. The remote machine may run UNIX/Linux or Windows. At this point, this is more like a toy than a serious application, because LREADjav can pose security problems. In the default configuration, it can only be used from “localhost”, but it can be configured to allow connections from three different remote clients. As they are identified via their IP address only, there is no password protection or the like. However, if a user has a serious need for that, he can easily implement a login/ password scheme...it's all open source!

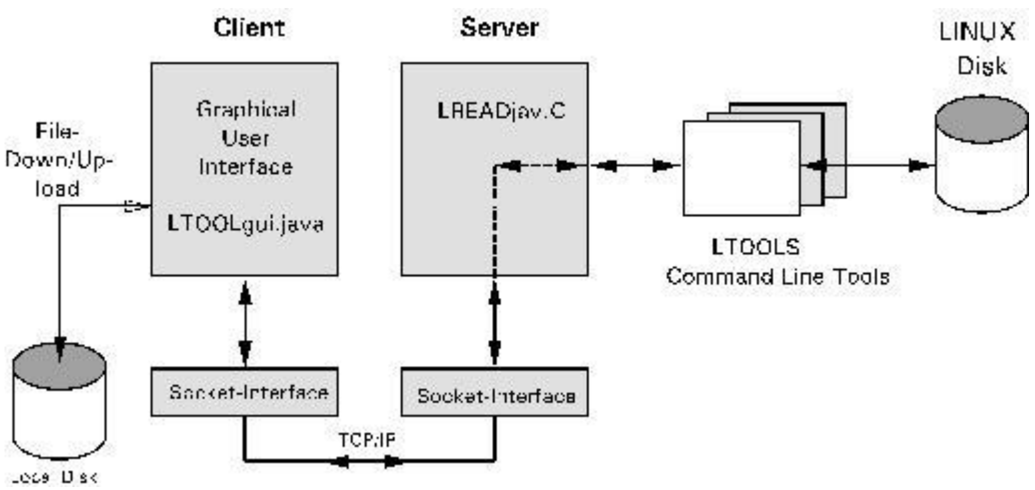
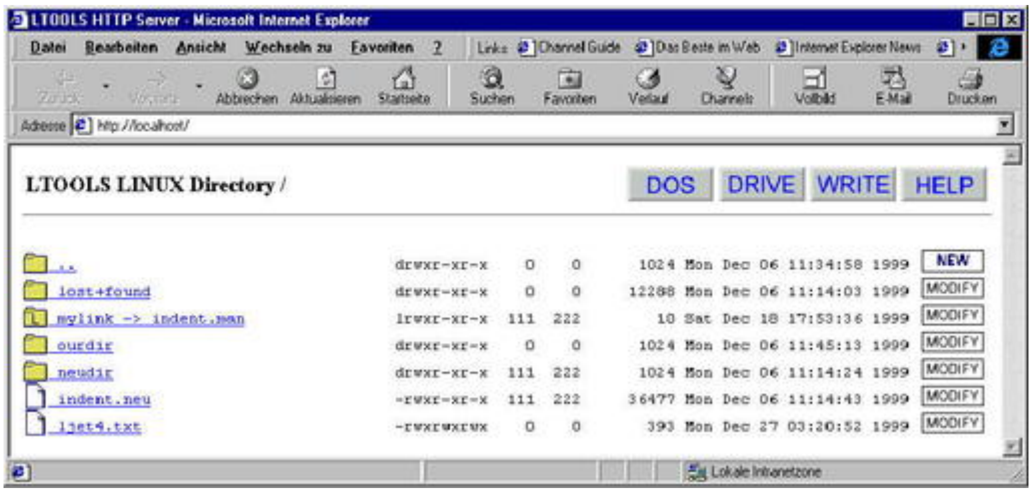


Figure 2. LTOOLgui for Remote Access

No Java? Use Your Web Browser

Maybe you don't have Java 2 installed. Well, no problem, as long as you have a web browser. Start **LREADsrv** and your web browser and as URL type **http:localhost** (see Figure 3). Now, your Linux directory listing should show up graphically in your web browser. LREADsrv is a small local web server, that, via a simple CGI-like interface, makes the LTOOLS accessible via HTTP-requests and converts their output dynamically into HTML pages (see Figure 4). Of course, not only does this provide local access, but it also allows remote access via the Internet. However, for remote users, LREADsrv does have the same low level of security as LREADjav.

Because LREADsrv is based on HTML forms, which do not support drag-and-drop or direct copy-and-paste, working with your web browser is a little less convenient than working with the Java based GUI. Nevertheless it provides the same features.


LTOOLS HTTP Server - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Wechseln zu Favoriten ?
Links Channel Guide Das Beste im Web Internet Explorer News
Zurück Vorwärts Abbrechen Aktualisieren Startseite Suchen Favoriten Verlauf Channels Vollbild E-Mail Drucken
Adresse http://localhost/
DOS DRIVE WRITE HELP
Lokale Intranetzone

LTOOLS LINUX Directory /

```
..                      drwxr-xr-x    0    0     1024 Mon Dec 06 11:34:58 1999  NEW
lost+found              drwxr-xr-x    0    0    12288 Mon Dec 06 11:14:03 1999  MODIFY
mylink -> indent.man    lrwxr-xr-x  111  222       10 Sat Dec 18 17:53:36 1999  MODIFY
ourdir                  drwxr-xr-x    0    0     1024 Mon Dec 06 11:45:13 1999  MODIFY
neudir                  drwxr-xr-x  111  222     1024 Mon Dec 06 11:14:24 1999  MODIFY
indent.neu              -rwxr-xr-x  111  222    36477 Mon Dec 06 11:14:43 1999  MODIFY
ljet4.txt               -rwxrwxrwx    0    0      393 Mon Dec 27 03:20:52 1999  MODIFY
```

Figure 3. Exploring Linux Files with Microsoft's Internet Explorer

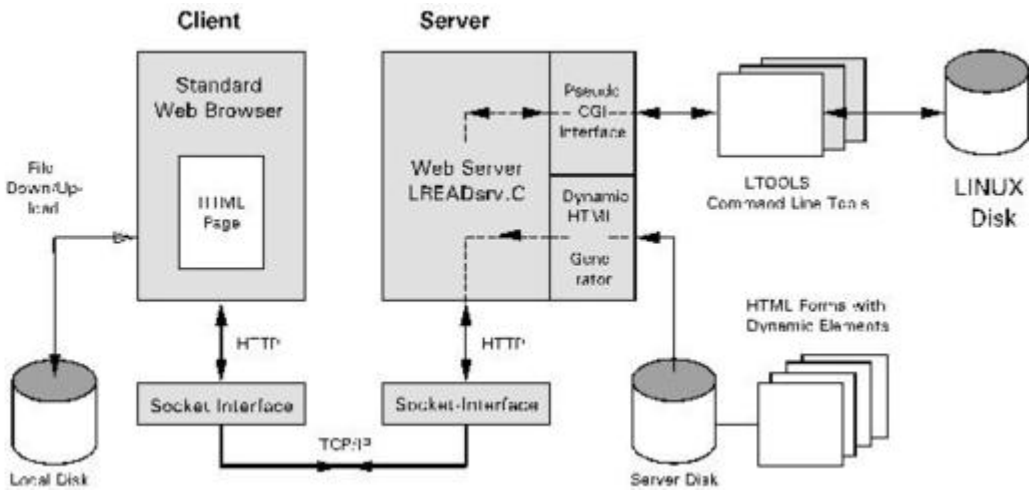


Figure 4. LREADsrv—HTTP-Based Access to Linux files

LTOOLS Internals—Accessing the Hard Disk under Windows

As DOS/Windows itself does not support interfaces to foreign file systems, the LTOOLS must access the “raw” data bytes directly on the disk. To understand the internals of the LTOOLS, you need to have a basic understanding of the following areas:

- How hard disks are organized in partitions and sectors and how they can be accessed, i.e., how “raw” bytes can be read or written from disk.
- Where this information can be found.
- How Linux's Extended two-file system is organized.

This automatically leads to the layered architecture of the LTOOLS kernel (see Figure 5), which consists of several C files:

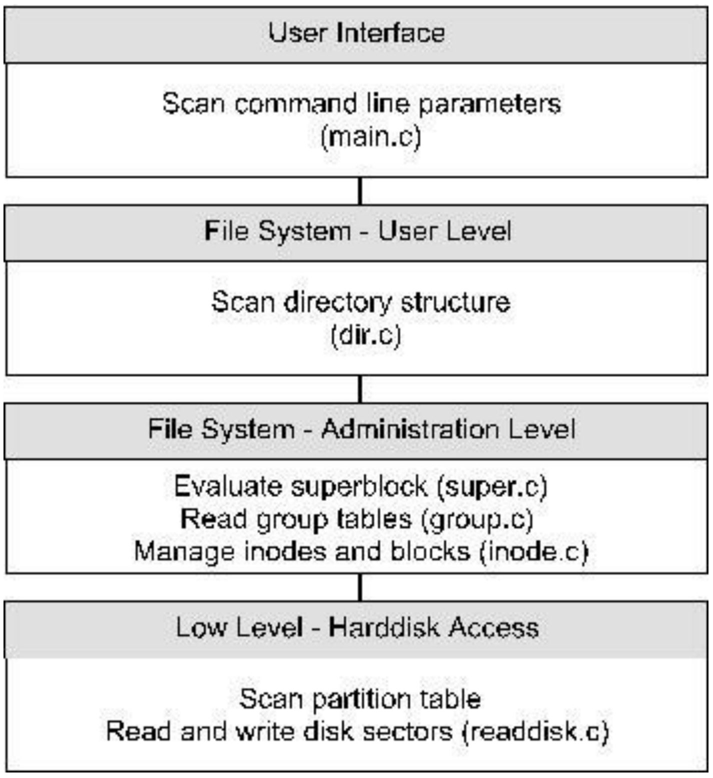

User Interface
Scan command line parameters (main.c)
File System - User Level
Scan directory structure (dir.c)
File System - Administration Level
Evaluate superblock (super.c)
Read group tables (group.c)
Manage inodes and blocks (inode.c)
Low Level - Harddisk Access
Scan partition table
Read and write disk sectors (readdisk.c)

Figure 5. LTOOLS Layered Architecture

- The lowest layer, layer 1 (in file Readdisk.c), physically accesses the hard disk. This layer deals with (nearly all) differences between DOS, Windows 9x, Windows NT and Linux/UNIX concerning direct hard disk access and tries to hide them from the higher layers (more about that soon).
- Layer 2 deals with the UNIX typical inode, block and group structures, into which the ext2 file system is organized.
- Layer 3 manages the directory structure of the file system.
- The highest layer 4 (in Main.c) provides the user interface and scans the command-line parameters.

By scanning your hard disk's partition table, the LTOOLS try to automatically find the first Linux partition on the first hard disk. If you want to access another partition or disk, you have to specify it by the command-line parameter **-s**, e.g., **-s/dev/hdb2**. Alternatively, you may set another default drive and partition via command **ldrive**. To find out which partitions you have, call **ldir -part**.

Life was easy in the good old days of DOS. There was only one way for low-level read or write access to your hard disk: BIOS interrupt 13h /3/. BIOS data structures limited hard disks to 1,024 cylinders, 63 heads and 255 sectors of 512 bytes, or 8GB. Most C compilers provided a function named **biosdisk()**. This function could be directly used without needing to code in assembly language. Some years ago, in order to accommodate bigger hard disks, “extended” **int 13h** functions were introduced. To overcome the BIOS limitations, these functions used a linear addressing scheme, logical block addresses (LBA), rather than the old cylinder-head-sector (CHS) addressing.

This still works in Windows 9x's DOS window (see sidebar) as long as the program is compiled with a 16-bit compiler, at least for read access. (The LTOOLS use Borland C, the Windows NT version compiles with Microsoft Visual C, the UNIX/Linux version uses GNU C.) If you want low-level write access, you need “volume locks” /3/. This mechanism informs the operating system that your program is performing direct disk writes bypassing the operating system drivers, so that Windows can prevent other programs from accessing the disk until you're done. Again, this can be done without assembly programming by using the C compiler's **ioctl()** function.

<u>Low Level Hard Disk Access</u>

In a 16bit Windows program, BIOS functions can only be called via DPMI. As most C Compilers do not provide wrapper functions, this would require an in-line assembler. However, Win16 does not allow command line programs at all, so don't worry.

In Windows NT's DOS box, using BIOS int 13h will lead to a GPF (General Protection Fault). Due to safety reasons, Windows NT does not allow direct hard disk access bypassing the operating system. However, Microsoft provides a solution that is nearly as simple as what you would write under UNIX/Linux:

```
int disk_fd = open("/dev/hda1", O_RDWR);
```

This would open your hard disk's partition /dev/hda1; to read, you would call **read()**, to write you would call **write()**. Simple and straightforward, isn't it? Under Windows NT, if you use the WIN32 API /5/, function **CreateFile()** does not only allow the creation and opening of files, but also disk partitions:

```
HANDLE hPhysicalDrive =<\n>
      CreateFile("\\\\.\\PhysicalDrive0",
      GENERIC_READ | GENERIC_WRITE,
      FILE_SHARE_READ | FILE_SHARE_WRITE,
      0, OPEN_EXISTING, 0, 0 );
```

Reading and writing disk sectors can now be done via **ReadFile()** and **WriteFile()**.

You might think that you could use the same Win32 function under Windows 9x. But, if you read on in the documentation for **CreateFile()**, you will find:

```
Windows 95: This technique does not work for opening<\n>
a logical drive. In Windows 95, specifying a string in this form causes CreateFile to return an error.
```

Under Windows 9x, Microsoft's Win32 documentation recommends to call BIOS Int 13h via VWIN32, one of the system's VxDs (kernel drivers). However, if you try to do so, you won't succeed. Problem report Q137176 in Microsoft's Knowledge Base states that, despite what the official Win32 documentation says, this only works for floppy disks, not hard disks. As the problem report says, for hard disks, the only option is to call BIOS Int 16h in 16bit code. To call 16bit code from a 32bit program, you need Microsoft's “32bit to 16bit thunking”. This is not only another API (with other undocumented features or documented bugs?), thunking also requires Microsoft's thunking compiler, which from a definition script generates assembler code. From that, a 16bit and a 32bit object file must be generated using Microsoft's assembler MASM. These will be linked with some dozen lines of C-code, which you have to write, resulting in a 16bit and a 32bit DLL (dynamic link library). By the way, you need not only 32bit Visual C++ for this, but you must also have an old 16bit version of Microsoft's C compiler. Using a bundle of proprietary, not widely used tools is not a good solution for an open-source software tool like the LTOOLS!

In summary, there must be separate versions for DOS/Windows 9x, Windows NT and Linux/UNIX. To hide this from the user as far as possible, LTOOLS tries to find out under which operating system it is running and automatically calls the appropriate executable.

Safety Concerns

Using the LTOOLS may, to a certain extent, pose security problems. Any user may access and modify files on the Linux file system; change file access rights or file owners; exchange password files, and so on. However, this is possible with a simple disk editor, too. Nevertheless, unlimited access is only possible if running DOS or Windows 9x. Under Windows NT, the LTOOLS user needs to have administration rights to access the hard disk directly. In most standard installations of UNIX/Linux, only the system administrator has access rights for the raw disk devices /dev/hda, /dev/hda1, etc.

Are There Alternatives?

The LTOOLS are not the only solution for accessing Linux files from DOS/ Windows. Probably, Claus Tondering's Ext2tool /6/, a set of command line tools developed in 1996, was the first solution for this problem. However, Ext2tool is restricted to read-only access and does not run under Windows NT. Based on the Ext2tool, in 1997, Peter Joot wrote a windows NT version, still limited to read only /7/. Both programs were written in C and source codes are available.

John Newbigin provides us with Explore2fs /8/, which comes with a very nice GUI and runs under Windows 9x and Windows NT. With read and write access, it provides the same features as LTOOLgui. By the way, John has done great work; he even managed to implement Microsoft's 32bit to 16bit thunking (see above) under Borland's Delphi! All Delphi programs Explore2fs integrate 'seamlessly' into Windows, but porting to non-Windows operating systems may be difficult.

History and Future

The first version of the LTOOLS, with the original name “lread”, was created by Jason Hunter and David Lutz at Willamette University in Salem, Oregon. This first version ran under DOS, could show Linux directory listings and copy files from Linux to DOS, and was limited to small IDE hard disks and Linux on primary partitions.

I took over maintenance and further development in 1996. Since then, the LTOOLS have learned to deal with bigger hard disks, access SCSI drives, and run under Windows 9x and Windows NT. They have additional write access and were ported back to UNIX run under Solaris and Linux itself. They now have a web browser-based and a Java-based graphical user interface, etc. Many Linux users, most of them named in the source code, helped in testing and debugging. Thank you.

In the meantime, LTOOLS reached version V4.7 /1/ at the time this article was written. Besides additional features, a lot of bugs have been fixed—and most likely new ones have been introduced. A common problem has persisted over the years: Nobody foresaw the rapid advances in hard disk technology, where disk sizes have exploded and consistently hit operating system limits. Do you remember DOS's problems with 512MB disks, Windows 3.x problems with 2GB partitions, BIOS's limit at 8GB and the various problems that Windows NT had at 2GB, 4GB and 8GB? It was only a short time ago. And, by the way, even Linux has its problems. In kernels previous to 2.3, no file could exceed 2GB, as Linux, like most 32bit UNIX systems, uses a signed 32bit offset pointer in read() or write(). (This is resolved in kernel 2.4 by changing offsets to 64bit values, but maintaining upward compatibility may drive Linux into the same problems that we discussed for Windows above.) Software standardization for disk access has always occurred much slower than the disk developers pace, so they invented proprietary solutions to overcome the operating system limits. As always, the developers of LTOOLS—and many other programmers—had to deal with it. So don't be angry if the LTOOLS don't work for you on your brand new 64GB drive. It's open source, so simply try to help debug and further develop them.

Don't forget, if you use the LTOOLS, you do so at your own risk! Read-only access to Linux is not critical. However, if you use write access to delete files or modify file attributes on your Linux disk, the LTOOLS—and you as the user—can create a real mess, so always keep a backup.

Resources

In “real life”, **Werner Zimmermann** (Werner.Zimmermann@fht-esslingen.de) teaches control engineering, digital systems and computer architecture at the FH Esslingen—University of Applied Sciences, Esslingen, Germany. He has a hardware and software background in automotive and industrial embedded systems. His “career” as a Linux system software developer started in 1994, when he purchased a CD-ROM drive, which was not supported by Linux. He developed aztcd.c, a Linux CD-ROM driver, which is still included in all standard Linux kernels, even if the drive now is very much outdated (http://www/it.fht-esslingen.de/~zimmerma/).

Archive Index Issue Table of Contents

Advanced search

A Web-Based Lunch Ordering System

Cheng-Chai Ang

Issue #79, November 2000

The author demonstrates how easy it is to write in Python—and make sure you get steamed, not fried rice.

Maybe this article should be entitled “How I Discovered Python and Ditched Everything Else”, instead. I have always wanted to write web-based applications but somehow found that getting started was quite intimidating. So, having procrastinated for years, I finally got around to writing my first application. My work required an intra-office application for which some values needed prompting on a web page. These values are sent to a CGI script, cross-verified via an SQL database, dispatched to a waiting process via sockets, and the results sent back to the web page.

By luck, I stumbled upon a scripting language called Python. I was reading a recent issue of *Linux Journal* (December 1999), in which they interviewed Eric Raymond, who mentioned that he now programs only in Python. At that point, I was a day into implementing the above system in Perl and was not quite finished. If Python was good enough for Eric, it was worth a try.

Well, I finished what I wanted to do in just over two hours. This was using a language that I had not heard of two hours earlier. At the risk of losing my professional advantage, I thought I would share with others how easy Python is to use, especially to do CGI (and almost everything else). As the above application would be too technical and boring to actually work through (and I'd probably get sued by my employers), I've decided to work through another, much more interesting exercise.

The Problem Description

Work being situated in a semi-remote location (culinary-wise, except for the place next door, which has excellent food but is a bit expensive to eat lunch at every day), take-out lunch was organised to be delivered to us once a week.

Each participant took turns organised the lunch orders. Being spread out over three floors, it was quite a chore, and no one looked forward to doing it. A web-based ordering system seemed to me the obvious solution but not having done any CGI programming before, it seemed quite overwhelming. The others did not seem to care. But writing CGI web systems can be quite simple, especially when one can do it using Python. (Okay, Perl gurus may disagree, but that's the whole point, i.e., a Perl guru versus a Python novice!)

The Initial Requirements

I knew roughly how I wanted it structured. There'd be a web page with a pull-down list with the restaurant menu, and, by clicking on a submit button, an e-mail with the person</#146>s order would be sent to the nominated lunch organiser.

Based on hearsay and some cursory research on the Net, I decided to use the following tools:

- Javascript for the client end (the web page)
- Python for the server side
- Apache for the web server, which is distributed with Linux (well, it was with my copy of Red Hat Linux 6.2); there is also a Windows version, too, if one is so inclined
- Linux for the web server OS

In designing the web pages, I decided to keep it fairly simple: a pull-down box with some radio buttons (see Figure 1).

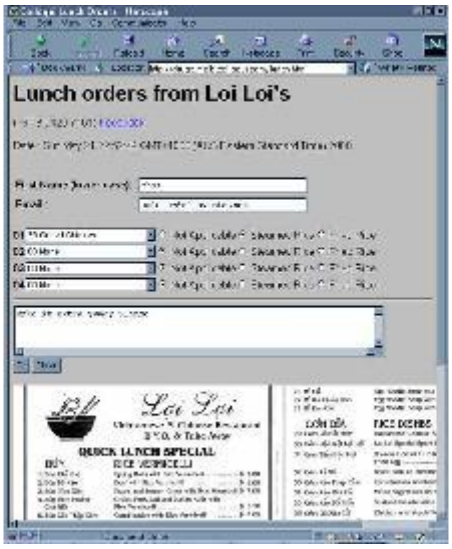


Figure 1. Snapshot of lunch.html

I could have used some HTML editor but decided that I could not handle learning another new package, so I did it by hand. Since what I wanted to do wasn't complicated, the by-hand method proved sufficient.

It was easy to install the web server using Linux. When I was installing Linux, the option to install Apache was ticked. When I typed in **localhost** as the URL to Netscape, it displayed the Apache page with the message that if I saw that page, everything was A-Okay! Whoo-hooo... so far so good. (See http://www.apache.org/ for more details). You'll probably need to be root (the superuser) to do the install.

Using Javascript to write a web page seems semi-obvious. There are several functions for data input verification (**ValidLength()** and **ValidEmail()**). **MenuHeader()** displays the header part of the page. Each call to MenuEntry() displays an input row. In this case, it is called four times, once for each lunch order item (see Listing 1).

Listing 1

The most tricky lines would be the **ON-SUBMIT** statement:

```
<FORM NAME="order" onSubmit="return Validate();"
    ACTION="http://localhost/cgi-bin/lunch.cgi"
METHOD="POST">
```

There are two ways a web page can communicate with a CGI script: **GET** and **POST**. In a nutshell, GET sends the information as part of the URL (i.e., you might have seen some URLs which resemble http://localhost/script.cgi?param1=value1&param2=value2 in your surfing of the Internet). When POST is used, the form information is sent via the standard input, i.e., the CGI script needs to read in standard input, and then parse the input separate out the various parameters.

It is generally accepted that POST is better (more robust, not limited by the maximum character limit of shell used). The methods used to extract the data differ according to whether POST or GET is used, but Python hides this from you (which is good).

Viewing lunch.html Via the Web Server

I then placed the lunch.html file in the directory /home/httpd/html:

```
$ cp lunch.html /home/httpd/html
```

(This is the default location Apache looks for html files. It can be configured to look elsewhere.) Once you have done this, you can see what lunch.html looks like by browsing http://localhost/lunch.html using Netscape (or any browser).

Write the CGI script with Python, which is distributed with the Red Hat Linux distribution (see http://www.python.org/). After consulting the Python documentation, which also came with the system, my first script looked like the one shown in Listing 3. It is essentially a cannibalised version of an example found in the Python documentation. To make use of this script, you'll need to point the CGI script specified in the ACTION statement in the HTML file to this script instead. That is, change the cgi script specified in the ACTION statement from lunch.cgi to first.cgi.

Testing the Installation

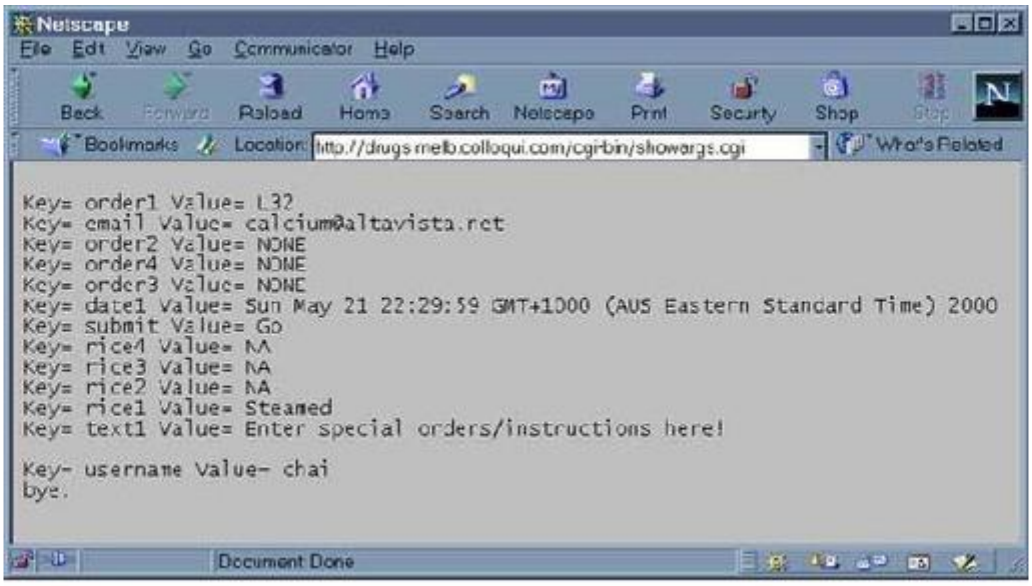
I then copied first.cgi to the directory /home/httpd/cgi-bin:

```
$ cp first.cgi /home/httpd/cgi-bin
```

Essentially, I interrogated all the variables sent to the script by the form and printed it back out. All output printed will be displayed by the browser.

```
#-----------------
#!/usr/bin/env python
# first.cgi
import cgi                                   # import the cgi module

print "Content-Type: text/plain\n\n"       # necessary for the browser

lunchForm = cgi.FieldStorage()               # retrieve the values

for name in lunchForm.keys():
    print "Key= " + name + " Value= " + lunchForm[name].value + " "

print "bye."
#-------------
```

When the Go button on the lunch.html page is clicked and the first.cgi script is activated, the output returned to the web browser looks like that shown in Figure 2.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://drugs.melb.colloqui.com/cgi-bin/showargs.cgi What's Related
Document Done

```
Key= order1 Value= L32
Key= email Value= calcium@altavista.net
Key= order2 Value= NONE
Key= order4 Value= NONE
Key= order3 Value= NONE
Key= date1 Value= Sun May 21 22:29:59 GMT+1000 (AUS Eastern Standard Time) 2000
Key= submit Value= Go
Key= rice4 Value= NA
Key= rice3 Value= NA
Key= rice2 Value= NA
Key= rice1 Value= Steamed
Key= text1 Value= Enter special orders/instructions here!

Key= username Value= chai
bye.
```

Figure 2. Output of Script first.cgi

You will notice that the keys found in the CGI script correspond to the variables I used in lunch.html.

Once I got this simple script working, I then expanded it to do what I wanted (see Listing 3). The Python code is quite straightforward and self-explanatory. It imports the CGI module, then calls the member function **FieldStorage()** of CGI. Whether the information is sent using the GET or POST method is hidden from you. That's how all the information sent by the web page is retrieved. The information can then be extracted by accessing lunchForm.

Listing 2

The body of the mail sent is then constructed via a series of writes to sendmail, a UNIX sendmail mail transfer agent. I decided to mail the lunch order to user lunch@localhost. An alias can be inserted in file /etc/aliases:

```
lunch:          chai@localhost
```

where user chai@localhost is organising the orders. This way, if the lunch organiser gets changed, the file /etc/aliases needs to be changed and not the CGI script. (**newaliases** needs to be run for changes to /etc/aliases to take effect).

Easy, eh? Well, it could be much worse.

Hooking Everything Together

I then copied lunch.cgi to the directory /home/httpd/cgi-bin:

```
$ cp lunch.cgi /home/httpd/cgi-bin
```

I opened Netscape, typed in http://localhost/lunch.html as the URL, filled in the form, selected my order and clicked on “Go”.

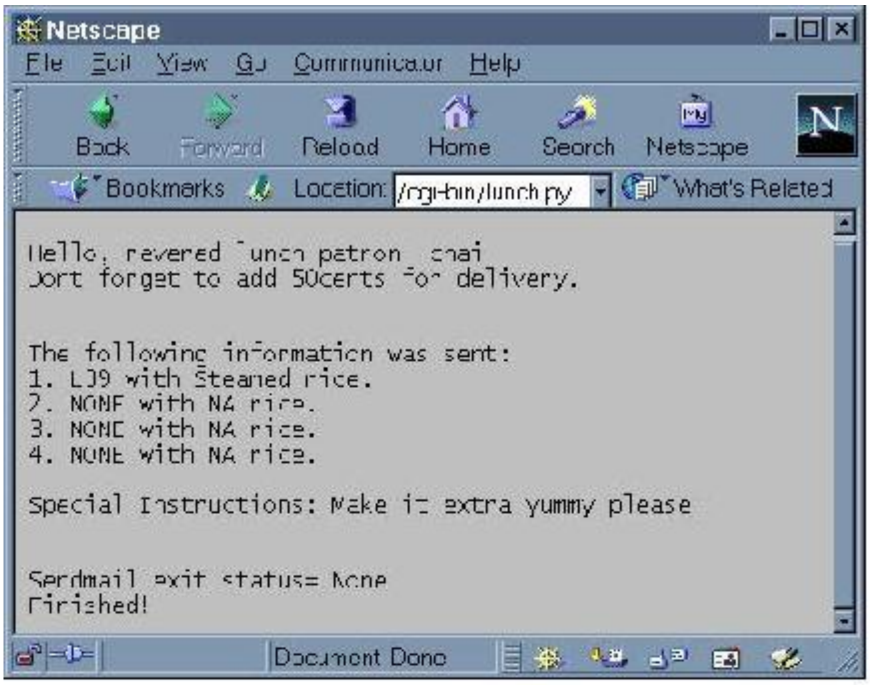
Sometime later, an e-mail arrived outlining the order.

Here is what the received e-mail of the lunch order, sent by the CGI script, looks like:

```
>From nobody@localhost  Wed Apr 26 11:01:50 2000<\n>
Delivered-To: ccang@localhost
Date: Wed, 26 Apr 2000 11:01:48 +1000
To: lunch@localhost
From: chai <calcium@altavista.net>
Subject: loi loi
Sender: nobody@localhost
SourceIP 194.118.1.1
calcium@altavista.net
Wed Apr 26 11:01:01 GMT+1000 (EST) 2000
chai wants
1. L39 with Steamed rice.
2. NONE with NA rice.
3. NONE with NA rice.
4. NONE with NA rice.
```

Special Instructions: Make It Extra Yummy Please!

Figure 3 shows what the web page looks like after the request has been sent.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape
Bookmarks Location: /cgi-bin/lunch.py What's Related
Document Done

```
Hello, revered lunch patron  chai
Dont forget to add 50cents for delivery.


The following information was sent:
1. L39 with Steamed rice.
2. NONE with NA rice.
3. NONE with NA rice.
4. NONE with NA rice.

Special Instructions: Make it extra yummy please


Sendmail exit status= None
Finished!
```

Figure 3. Snapshot of Resulting Web Page After Lunch Order Made

Conclusion

Is the on-line system better than write-order-on-scrap-paper method? Debatable, but it certainly is more fun (at least for me).

Improvements? The web page is geared towards an individual making an order, as opposed to a person ordering for multiple people. In hindsight, the web page could have been laid out with the latter in mind, and, being a superset of the former, would satisfy those requirements as well. A simple compromise could be having a multiples box, which would allow a person to order more than one serving of the same dish per order row. In the current scheme, this would still only allow four different orders per e-mail. So make it 10? 20? How long is a piece of rope? (paraphrased to make it more sinister). A design problem left to the reader.

I suppose I could also hook it into an SQL database (http://www.mysql.org/) and print out a histogram of the past orders for a particular person. A by-product of using a database is that one could print out reports, e.g., what is to be ordered for that week.

I suppose if there is enough interest and if I have enough time, I'll add a second part to this system, where the CGI script would interact with a SQL database and return HTML code to display a frequency list. And, perhaps, with some cookie interaction.

Finally, on a personal note, I've seen the future and it is Python. Look it up (and JPython too!).

Resources

Thanks to Python's simplicity, **Cheng-Chai Ang** (calcium@ozemail.com.au) has blossomed from a novice Python programmer to a novice Python programmer doing useful (sometimes nontrivial) stuff almost instantly. He works for Carbonated Software Pty, Ltd. and recently started doing JSP/Java servlets after ten years of C++.

Advanced search

Bare Metal Recovery

Charles Curley

Issue #79, November 2000

Most us don't take the time to plan for disaster recovery. One excuse is not wanting to figure out what to do. One excuse down—this article gives you the step-by-step.

Imagine your disk drive has just become a very expensive hockey puck. Imagine you have had a fire, and your computer case now looks like something Salvador Dalí would like to paint. Now what?

Bare metal recovery is the process of rebuilding a computer after a catastrophic failure. This article is a step-by-step tutorial on how to back up a Linux computer to be able to make a bare metal recovery, and how to make that bare metal recovery.

The normal bare metal restoration process is: install the operating system from the product disks, install the backup software (so you can restore your data), and then restore your data. Then, you get to restore functionality by verifying your configuration files, permissions, etc.

The process here will save installing the operating system product disk. It will also restore only the files that were backed up from the production computer, so your configuration will be intact when you restore the system. This should save you hours of verifying configurations and data.

The target computer for this article is a Pentium computer with a Red Hat 5.2 Linux server installation on one IDE hard drive. It does not have vast amounts of data because the computer was set up as a “sacrificial” test bed. That is, I did not want to test this process with a production computer and production data. Also, I did a fresh “server” install before I started the testing so that I could always reinstall if I needed to revert to a known configuration.

The target computer does not have any other operating systems on it. While it simplifies the exercise at hand, it also means if you have a dual boot system, you will have to experiment to get the non-Linux OS to restore.

The process shown below is not easy. Practice it before you need it! Do as I did, and practice on a sacrificial computer.

Nota Bene: The sample commands will show, in most cases, what I had to type to recover the target system. You may have to use similar commands, but with different parameters. For example, below we show how to make a swap device on /dev/hda9. It is up to you to be sure you duplicate your setup, and not the test computer's setup.

The basic procedure is set out by W. Curtis Preston in *Unix Backup & Recovery*, (http://www.ora.com/, http://www.oreilly.com/catalog/unixbr/), which I favorably reviewed in *Linux Journal*, October 2000. However, the book is a bit thin on the ground. For example, exactly which files do you back up? What metadata do you need to preserve, and how?

We will start with the assumption that you have backed up your system with a typical backup tool such as Amanda, Bru, **tar**, Arkeia or **cpio**. The question, then, is how to get from toasted hardware to the point where you can run the restoration tool that will restore your data.

Users of Red Hat Package Manager (RPM)-based Linux distributions should also save RPM metadata as part of their normal backups. Something like:

```
rpm -Va > /etc/rpmVa.txt
```

in your backup script will give you a basis for comparison after a bare metal restoration.

To get to this point, you need to have:

- Your hardware up and running again, with replacement components as needed. The BIOS should be correctly configured, including time, date and hard drive parameters.
- A parallel port Iomega Zip drive or equivalent. You will need at least 30MB of space.
- Your backup media.
- A minimal Linux system that will allow you to run the restoration software.

To get there, you need at least two stages of backup, and possibly three. Exactly what you back up and in which stage you back it up is determined by your restoration process. For example, if you are restoring a tape server, you may not need networking during the restoration process, so only back up networking in your regular backups.

You will restore in stages as well. In stage one, we build partitions, file systems, etc., and restore a minimal file system from the Zip disk. The goal of stage one is to be able to boot a running computer with a network connection, tape drives, restoration software or whatever we need for stage two.

The second stage, if it is necessary, consists of restoring backup software and any relevant databases. For example, suppose you use Arkeia and build a bare metal recovery Zip disk for your backup server. Arkeia keeps a huge database on the server's hard drives. You can recover the database from the tapes, if you want. Instead, why not tar and **gzip** the whole Arkeia directory (at /usr/knox), and save that to another computer over nfs? Stage one, as we have defined it, does not include X, so you will have some experimenting to do to back up X as well as your backup program.

Of course, if you are using some other backup program, you may have some work to do to. You will have to find out which directories and files it needs to run. If you use tar, gzip, cpio, **mt** or **dd** for your backup and recovery tools, they will be saved to and restored from our Zip disk as part of the stage one process described below.

The last stage is a total restoration from tape or other media. After you have done that last stage, you should be able to boot to a fully restored and operational system.

Preparation

First, do your normal backups on their regular schedule. This article is useless if you don't do that.

Next, build yourself a rescue disk. I use **tomsrtbt**, available at http://www.toms.net/~toehser/rb/. It is well documented and packs a lot of useful tools onto one floppy diskette. There is an active list for it, and the one question I had was quickly and accurately answered. I like that in a product my shop may depend on one day.

Next, figure out how to do the operating system backup you will need so that you can restore your normal backup. I followed Preston's advice and used an Iomega parallel port Zip drive. They get approximately 90MB of useful storage to a disk. Since the scripts I developed for the stage one backup save about 30MB of data, one Zip disk should be plenty for the job at hand.

Installing the Zip Drive

Much of this is covered in the Zip Drive HOWTO (www.linuxdoc.org/HOWTO/mini/ZIP-Drive.html), so I'll show you exactly what I did. Your mileage may vary. You may have already done a lot of this, in which case your setup may vary. For me the procedure was simplified by not having a printer share the parallel port with the Zip drive.

First, install the driver for the Zip drive, and make a mount point for it:

```
[root@tester /etc]# modprobe ppa
[root@tester /etc]# mkdir /mnt/zip
```

Insert a suitable line into your fstab:

```
/dev/sda4       /mnt/zip        vfat    noauto  0 0
```

Save fstab. Put a Zip disk in the drive. Then you should be able to mount the Zip drive:

```
[root@tester /etc]# mount /mnt/zip
[root@tester /etc]# ls -l /mnt/zip
total 277
drwxr-xr-x  2 root   root    16384 Dec 31  1969 .
drwxr-xr-x  7 root   root     1024 May 11 08:34 ..
-rwxr-xr-x  1 root   root   265728 Jan 30 1998 50ways.exe
```

Putting ex2fs on the Zip Disk

Zip drives come formatted for MS-DOS or Windows (FAT) or the Mac. The FAT format is somewhat inefficient for what we are doing, although not fatally so. Our test computer setup put about 26MB onto the Zip disk, so you can skip installing the ext2fs on your Zip disk.

Here is how to replace the FAT file system on the Zip disk with an ext2fs. First, unmount the Zip drive:

```
[root@tester /etc]# umount /mnt/zip
```

Then run fdisk and see what you have:

```
[root@tester /etc]# fdisk /dev/sda

Command (m for help): p

Disk /dev/sda: 64 heads, 32 sectors, 96 cylinders
Units = cylinders of 2048 * 512 bytes

Device Boot     Start    End      Blocks   Id      System
/dev/sda4  *    1        96       98288    6       DOS 16-bit >=32M

Command (m for help):
```

For reasons known only to Murphy and Iomega, FAT Iomega Zip disks have only one partition, partition four. Delete the offending partition:

```
Command (m for help): d
Partition number (1-4): 4

Command (m for help): p

Disk /dev/sda: 64 heads, 32 sectors, 96 cylinders
Units = cylinders of 2048 * 512 bytes

Device Boot     Start   End     Blocks  Id      System
```

Per the Zip drive HOWTO, we will make a new partition as partition 1:

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-96): 1
Last cylinder or +size or +sizeM or +sizeK  ([1]-96):  96
```

Displaying the partition table indicates that it was marked as a Linux ext2fs partition for us, so we don't have to change the file system id:

```
Command (m for help): p

Disk /dev/sda: 64 heads, 32 sectors, 96 cylinders
Units = cylinders of 2048 * 512 bytes

Device Boot     Start   End     Blocks  Id      System
/dev/sda1       1       96      98288   83      Linux native
```

Use the **w** command to write the partition table and exit. We now have to make a file system on the freshly minted partition:

```
[root@tester /etc]# mke2fs /dev/sda1
mke2fs 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
Linux ext2 file system format
File system label=
24576 inodes, 98288 blocks
4914 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
12 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
  8193, 16385, 24577, 32769, 40961, 49153, 57345,
  65537, 73729, 81921, 90113

Writing inode tables: done
Writing superblocks and file system accounting information: done
```

Now we go back to fstab and add a new entry:

```
/dev/sda1    /mnt/zip    ext2   noauto    0 0
```

This lets us specify which file system we have on the Zip disk by mounting the device file rather than the mount point. For example:

```
[root@tester /etc]# mount /dev/sda1
```

The order in which you place the two lines in /etc/fstab is important. The first one determines whether the default partition mount will try to mount if you specify /mnt/zip. So put this line above the entry you made earlier. We will use this in the script that saves the stage one data to the Zip disk.

Creating the Stage One Backup

Having made your production backups, what additional information do you need to back up in order to rebuild your system? You need to preserve your partition information so that you can rebuild them. This and other metadata are preserved in the script **save.metadata** (see Listing 1).

Listing 1. save.metadata

Unfortunately, fdisk does not yet export partition information in a manner that allows you to reimport it from a file. Since you have to rebuild your partitions by hand using fdisk, we will save it as a human-readable text file.

Fortunately, the price of hard drives is plummeting almost as fast as the public's trust in politicians after an election. So it is good that the hand-editing process allows the flexibility necessary to use a larger replacement drive.

The script saves the partition information in the file fdisk.hda in the root of the Zip disk. It is a good idea to print this file and your /etc/fstab. Then, you can work from hard copy while you restore the partition data. You can save a tree by toggling between two virtual consoles, running fdisk in one and catting /etc/fstab or /fdisk.hda as needed, but this strikes me as error prone.

You will also want to preserve files relevant to your restoration method. For example, if you use nfs to save your data, you will need to preserve hosts.allow, hosts.deny, exports, etc. Also, if you are using any network backed restoration process, such as Amanda or Quick Restore, you will need to preserve networking files like **HOSTNAME**, hosts, etc. and the relevant software tree.

The simplest way to handle these and similar questions is to preserve the entire /etc directory.

Preston cheats when he backs up his system. There is no way a 100MB Zip drive is going to hold a server installation of Red Hat 5.2. A 250MB Zip disk will hold a fresh server installation, but probably won't hold a production server. We have to be much more selective than simply preserving the whole kazoo. What files do we need?

- The boot directory.
- The /etc directory and subdirectories.
- Directories needed at boot time.

- Device files in /dev.

To determine the directories needed at boot, we look at the boot initialization file **/etc/rc.sysinit**. It sets its path like so:

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH
```

Trial and error indicated that we needed some other directories as well, such as /dev. In retrospect, of course we need /dev. In Linux, you can't do much without device files.

In reading the script save.metadata (see Listing 1), again note that we aren't necessarily saving files that are called with absolute paths. We may require several iterations of backup, test the bare metal restore, reinstall from CD and try again, before we have a working backup script. While I worked on this article, I made five such interations before I had a successful restoration. That is one reason why it is essential to use scripts whenever possible. Test thoroughly!

Booting tomsrtbt

The first thing to do before starting the restoration process is to verify that the hardware time is set correctly. Use the BIOS setup for this. How close to exact you have to set the time depends on your applications. For restoration, within a few minutes of exact time should be accurate enough. This will allow time-critical events to pick up where they left off when you finally launch the restored system.

Before booting tomsrtbt, make sure your Zip drive is placed on a parallel port, either /dev/lp0 or /dev/lp1. The startup software will load the parallel port Zip drive driver for you.

I have one of those ne2000 clone Ethernet cards in my test system. This, it turns out, gives the 3c59x driver in the tomsrtbt kernel fits. The workaround is to tell the kernel to ignore its address range. At the LILO prompt:

```
zImage reserve=0x300,32
```

The next step is to set the video mode. I usually like to see as much on the screen as I can. So when the option to select a video mode comes, I use mode 6, 80 columns by 60 lines. Your hardware may or may not be able to handle high resolutions like that, so experiment with it.

Once tomsrtbt has booted and you have a console, mount the Zip drive. It is probably a good idea to mount it read only:

```
# mount /dev/sda1 /mnt -o ro
```

Check to be sure it is there:

```
# ls -l /mnt
```

Then clean out the first two sectors of the hard drive:

```
# dd if=/dev/zero of=/dev/hda bs=512 count=2
```

This sets the master boot record (MBR) to all zeros. It wipes out all record of the partitions and any boot code, such as LILO.

Then, using the hard copy of fdisk.hda you made earlier, use fdisk to partition the hard drive. Make the first cylinder a primary partition. After that, you will build one or more extended partitions with one to four logical partitions. Primary and extended partitions are numbered one to four. Partitions five and up are logical partitions. Notice also that the extended partitions overlap the logical partitions they contain. The partition number of a partition is the last one or two numbers in the device name, so /dev/hda5 is partition number five on hda.

You can have more than one primary partition, but this is unwise. Each primary partition precludes an extended partition, which can be subdivided into logical partitions. Extended partitions are far more flexible:

```
# fdisk

Command (m for help): p

Disk /dev/hda: 64 heads, 63 sectors, 1023 cylinders
Units = cylinders of 4032 * 512 bytes

Device Boot     Start   End     Blocks  Id      System

Command (m for help):
```

We will make a new partition as partition 1:

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1023): 1
Last cylinder or +size or +sizeM or +sizeK (1-1023): 9
```

Now we will make the extended partition:

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
Partition number (1-4): 2
First cylinder (1-1023): 10
Last cylinder or +size or +sizeM or +sizeK (10-1023): 1022
```

Note that the hard drive has 1,023 cylinders, and we only made the extended partition go to 1,022, wasting a cylinder. The reason is that we are duplicating the setup Red Hat gave us on the original installation.

Now for a logical partition:

```
Command (m for help): n
Command action
   l   Logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (10-1022): 10"
Last cylinder or +size or +sizeM or +sizeK (1-1022): 368
```

And so on for each partition that your fdisk.hda file indicates you should have.

Set the file system ID on your swap partition(s) to Linux Swap (82) using the t command.

Don't forget to use the a command to set a partition active, or bootable. In a Linux-only installation, this is usually the first partition, but can be another one. It will be the partition that mounts as /boot (see your printout of fstab):

```
Command (m for help): a
Partition number (1-9): 1
```

Then verify your work:

```
Command (m for help): v
372 unallocated sectors

Command (m for help): p

Disk /dev/hda: 64 heads, 63 sectors, 1022 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot  Start   End     Blocks  Id      System
/dev/hda1   *   1       9       18112+  83      Linux native
/dev/hda2       10      1022    2042208 5       Extended
/dev/hda5       10      368     723712+ 83      Linux native
/dev/hda6       369     727     723712+ 83      Linux native
/dev/hda7       728     858     264064+ 83      Linux native
/dev/hda8       859     989     264064+ 83      Linux native
/dev/hda9       990     1022    66496+  82      Linux swap

Command (m for help):
```

Check this listing against your printed copy. Note that the one extended partition overlaps the five Linux partitions.

Finally, we use the w command to write the partition table and exit:

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
 hda: hda1 hda2 < hda5 hda6 hda7 hda8 hda9 >
 hda: hda1 hda2 < hda5 hda6 hda7 hda8 hda9 >
Syncing disks.
```

```
WARNING: If you have created or modified any DOS 6.x partitions, please see
the fdisk manual page for additional information.
```

Then make ext2 file systems on each partition you will be using as an ext2 partition. These will be the primary and logical partitions that you did not change to swap partitions. Don't do this to your extended partitions!

```
mke2fs /dev/hda1
mke2fs /dev/hda5
 ...
```

For example,

```
# mke2fs /dev/hda1
mke2fs 1.10, 24-April-97 for EXT2 FS 0.5b, 95/08/09
Linux ext2 file system format
File-system label=
4536 inodes, 18112 blocks
905 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
3 block groups
8192 blocks per group, 8192 fragments per group
1512 inodes per group
Superblock backups stored on blocks:
        8193, 16385

Writing inode tables: done
Writing superblocks and file system accounting information: done
```

And so on for each primary or logical partition that is not a swap partition.

Now set up the swap partition.

```
# mkswap /dev/hda9
Setting up swap space, size = 68087808 bytes
```

Then, we have to manually build and mount a partition to each directory. Since what is now /target will eventually become /, we mount what will be / to /target:

```
# mkdir /target
# mount /dev/hda8 /target
```

Next, we build the directories we need in /target, and mount to them, like so for each directory:

```
# mkdir /target/boot
# mount /dev/hda1 /target/boot
```

You can determine which directories and mountspace points to build and mount from /mnt/etc/fstab. Fortunately, umask is already set correctly for almost all the directories we need to build.

And so on for each partition. Referring to /mnt/ls.root.txt, make sure you also set the proper modes for the directories you are building.

Do not try to mount the extended or swap partitions, though. You don't need to, and it won't do you any good, anyway.

To check your progress, use the command mount with no parameters.

```
# mount
/dev/ram0 on / type minix (rw)
none on /proc type proc (rw)
/dev/ram1 on /usr type minix (rw)
/dev/ram3 on /tmp type minix (rw)
/dev/sda1 on /mnt type ext2 (rw)
/dev/hda8 on /target type ext2 (rw)
/dev/hda1 on /target/boot type ext2 (rw)
/dev/hda6 on /target/home type ext2 (rw)
/dev/hda5 on /target/usr type ext2 (rw)
/dev/hda7 on /target/var type ext2 (rw)
```

Once you have created all your directories and mounted partitions to them, you can run the script /mnt/root.bin/restore.metadata (see Listing 2). This will restore the contents of the Zip drive to the hard drive.

Listing 2. Restore script! Zip to HD

You should see a directory of the Zip disk's root directory, then a list of the archive files as they are restored. **tar** on tomsrtbt will tell you that tar's block size is 20, and that's fine. You can ignore it. Be sure that LILO prints out its results:

```
Added linux *
```

That will be followed by the output from a **df -m** command.

If you normally boot directly to X, that could cause problems. To be safe, change your boot run level temporarily. In /etc/inittab, find the line that looks like this:

```
id:5:initdefault:
```

and change it to this:

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

Now, you can gracefully reboot. Remove the tomsrtbt floppy from your floppy drive if you haven't already done so, and give the computer the three-fingered salute, or its equivalent, “shutdown -r now”. The computer will shut down and reboot.

Second Stage Restoration

As the computer reboots, go back to the BIOS and verify that the clock is more or less correct.

Once you have verified the clock, exit the BIOS and reboot, this time to the hard drive. You will see a lot of error messages, mostly along the lines of “I can't find blah! Waahhh!” Well, if you have done your homework correctly up until now, those error messages won't matter. You don't need linuxconf or apache to do what you need to do.

You should be able to log into a root console (no X, no users, sorry). You should now be able to use the network, for example, to NFS mount the backup of your system.

If you did the two stage backup I suggested for Arkeia, you can restore Arkeia's database and executables. Now, you should be able to run **/etc/rc.d/init.d/arkeia start** and start the server. If you have the GUI installed on another computer with X installed, you should be able to log in to Arkeia on your tape server, and prepare your restoration.

When you restore, read the documentation for your restoration programs carefully. For example, tar does not normally restore certain characteristics of files, like suid bits. File permissions are set by the user's umask. To restore your files exactly as you saved them, use tar's p option. Similarly, make sure your restoration software will restore everything exactly as you saved it.

To restore the test computer:

```
[root@tester ~]# restore.all
```

If you used tar for your backup and restoration, and used the -k (keep old files, don't overwrite) option, you will see a lot of this:

```
tar: usr/sbin/rpcinfo: Could not create file:  File exists
tar: usr/sbin/zdump: Could not create file:  File exists
tar: usr/sbin/zic: Could not create file:  File exists
tar: usr/sbin/ab: Could not create file:  File exists
```

This is normal, as tar is refusing to overwrite files you restored during the first stage of restoration.

Just to be paranoid, run LILO after you perform your restoration. I doubt it is necessary, but if it is necessary, it's a lot easier than the alternative. You will notice I have it in my script, restore.all (see Listing 3).

Listing 3. restore.all Script

Now reboot. On the way down, you will see a lot of error messages, such as “no such pid.” This is a normal part of the process. The shutdown code is using the pid files from dæmons that were running when the backup was made to shut down dæmons that were not started on the last boot. Of course there's no such pid.

Your system should come up normally, with a lot fewer errors than it had before. The acid test of how well your restore works on an RPM based system is to verify all packages:

```
rpm -Va
```

Some files, such as configuration and log files, will have changed in the normal course of things, and you should be able to mentally filter those out of the report.

If you took my advice earlier and keep RPM metadata as a normal part of your backup process, you should be able to diff the two files, thereby speeding up this step considerably.

You should be up and running. It is time to test your applications, especially those that run as dæmons. The more sophisticated the application, the more testing you may need to do. If you have remote users, disable them from using the system, or make it “read only” while you test it. This is especially important for databases, to prevent making any corruption or data loss worse than it already might be.

If you normally boot to X, and disabled it above, test X before you re-enable it. Re-enable it by changing that one line in /etc/inittab back to: **id:5:initdefault:**

You should now be ready to rock and roll—and for some Aspirin and a couch.

Some Advice for Disaster Recovery

You should take your Zip disk for each computer and the printouts you made, and place them in a secure location in your shop. You should also store copies of these in your off-site storage location. The major purpose of off-site backup storage is to enable disaster recovery, and restoring each host onto replacement hardware is a part of disaster recovery.

You should also have several tomsrtbt floppies and possibly some Zip drives in your off-site storage as well. Have copies of the tomsrtbt distribution on several of your computers so that they back each other up. In addition, you should probably keep copies of this article, with your site-specific annotations on it, with your backups and in your off-site backup storage.

What Now?

This article is the result of experiments on one computer. No doubt you will find some other directories or files you need to back up in your first stage backup. I have not dealt with saving and restoring X on the first stage. Nor have I dealt with other operating systems in a dual boot system, or with processors other than Intel.

I would appreciate your feedback as you test and improve these scripts on your own computers. I also encourage vendors of backup software to document how to do a minimal backup of their products. I'd like to see the whole Linux community sleep just a little better at night.

Charles Curley (ccurley@trib.com) lives in Wyoming, where he rides horses and herds cattle, cats and electrons. Only the last of those pays well, so he also writes documentation for a small software company headquartered in Redmond, Washington.

Archive Index Issue Table of Contents

Advanced search

Software ICs

Robert D. Findlay

Issue #79, November 2000

“Complexity must be grown from simple systems that already work.” —Kevin Kelly

Software engineers should look to their hardware counterparts for approaches to managing complexity. Before the advent of Integrated Circuits, electronic circuits were generally complex creatures with many interconnected discrete components. The complexity of the circuit was visible, difficult to manage and adversely affected the cost of products. With the advent of ICs much of this complexity was hidden inside the chips themselves. The job of designing complex functionality into products became much easier.

Despite many attempts over the years, software design has never been able to replicate this IC design paradigm. The advent of object-oriented programming languages and tools was supposed to address some of these issues. While object-oriented design offers some significant improvements in areas such as GUI programming it doesn't always do a great job of hiding complexity. In fact, they often simply shift the complexity into other areas in the software development chain such as testing, toolsets, class library design or the learning curve.

Modern hardware design requires a complex skillset. What the advent of ICs did, however, was allow the hardware designer to use a given chip in a product design without having to understand the exact physics and layout of the circuits contained within the chip itself. As long as the designer conformed to the specs at the external interface to the chip (pins) the chip will react in a very predictable manner. This is what encapsulation of complexity offers: complexity hiding and predictable/reproducible behavior. Objects in software design offer partial complexity hiding, but the software designer often still has to know and master a complex language (e.g., C++) in order to be able to wire objects together into a product. Objects are very poor at providing predictable and reproducible software behavior. Furthermore, the language of the objects themselves often dictates the language used to “wire them together”. In our opinion, true encapsulation of complexity behind a universally simple and extendable API is what is required to produce a software IC. The software designer should not have to master a complex object-oriented language and toolset in order to be able to “wire together” these software ICs. At the very least the software designer should be able to choose the wiring language independent of the chip language.

Process Encapsulation

Many RTOSes have pioneered the use of user-space processes as an encapsulation scheme. One of the oldest to use this scheme is QNX TM (http://www.qnx.com/). Since 1980, QNX has released a continuous series of innovative OSes that were based upon a set of cooperating processes all using a Send/Receive/Reply messaging paradigm. QNX's approach to kernel design differs greatly from that used in Linux and we do not wish to reinflame the infamous microkernel vs. monolithic kernel debate. Suffice to say that we believe that the process model and the Send/Receive/Reply messaging paradigm first pioneered by QNX offers the key ingredients of a successful software IC.

In most modern operating systems, including Linux, the process is a very robust computing entity. There are several layers of protection offered to a process and the resources it embraces when running on a processor. It is very difficult for two processes to inadvertently adversely affect one another. If a given process encounters a fatal bug and crashes, it is rare that the entire system will be compromised. So the userispace process in Linux offers us a great container for our software IC. Now all we need is the software analog of the pins. This is where the messaging paradigm comes in.

Tokenized Messaging

One of the most powerful message formats is also one of the simplest. We can view a message as simply a collection of bytes. This collection of bytes is divided into two parts. The first field of bytes which is of fixed length is always present in every message and taken together, these bytes represent a unique message identifier called a token. The balance of the message (which can be of variable length) represents the token context-sensitive data. When two processes wish to exchange such messages they simply agree on a token scheme and on the format for each tokenized message they want to exchange. The interprocess message transport layer need not be concerned at all with the message content.

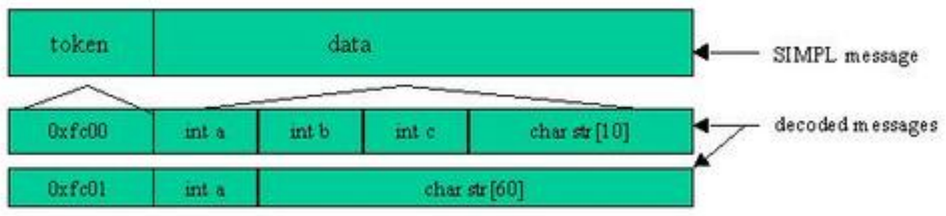


Figure 1. Tokenized

Message Synchronization

One of the goals of any software design is to produce software which behaves predictably and reproducibly under a given stimulus. Many object-oriented and message-oriented design paradigms suffer because they introduce a degree of unpredictability and randomness into a software product. If a message is exchanged with a mailbox scheme and the sender and the receiver are never brought into synchronization it will be exceedingly difficult to replicate or predict all possible states that two process system can be in. Much depends on the environment and timing considerations in which these two processes find themselves. These modules may work fine in one environment and suffer sporadic failures in another. This leads to increased testing and maintenance costs and a poorer software product. Often the blame is incorrectly directed at the multiprocess design paradigm. How often have we heard it said that client/server software design does not handle complexity well. The original designers of Send/Receive/Reply synchronized messaging believed that the answer lies in forcing a state-machine-like synchronization to occur on each message pass.

It works like this: a sending process composes a message and arranges to send it to a receiving process. While waiting for a reply the sending process is blocked. The receiving process, on the other hand, is held blocked until a message arrives destined for it. It unblocks, reads the message, processes its contents and then replies to the sender. This reply then unblocks the sender and the two processes are free to operate independently once again.

Many have argued that this blocking/forced synchronization introduces unnecessary complexity into a message exchange, but when properly applied it achieves exactly the opposite effect. By forcing synchronization to occur at each message pass, one finds that our multi-process applications now start to behave in a very predictable and reproducible manner. Gone are the timing and environmental effects that often plague nonsynchronized message passing schemes. When dealing with huge complex applications, this represents a huge strategic advantage. As an added bonus, since the sender is blocked during message transmission and is explicitly unblocked by the receiver process with the reply, it is very simple to arrange to transport these messages over a variety of media (including some which are “slow”). The messages could be exchanged via shared memory if the processes were on the same processor, or they could be exchanged via the Internet if the processes were physically separated or on a serial line in a dial up situation. While the performance of the collective of processes would obviously be affected by the message transmission speed, the predictability and reproducibility of that performance would not be.

The importance of predictability of software cannot be overstated when it comes to software testing. Nothing makes software QA people wish for a career change more than an application which exhibits unpredictable and unreproducible behavior.

SIMPL Open-Source Project

To help promote this software IC paradigm in the Linux community we started the SIMPL open-source project (www.holoweb.net/~simpl). SIMPL enabled processes are Linux processes with all the features and protections that affords. SIMPL enabled processes are able to exchange tokenized binary messages using a blocking Send/Receive/Reply messaging scheme. In short SIMPL processes have the makings of some great software ICs.

Before going into more detail on how these software ICs might be built, it is worth emphasizing the advantages of this model.

- In principle, SIMPL processes can be written in any language. While much of the code on the SIMPL web site is still written in C, there is no reason why a C++ or JAVA SIMPL process could not be created and talk transparently to another SIMPL process written in Tcl/Tk or Python.
- The language used to write the software IC itself in no way dictates the language of another software IC with which interaction takes place. Furthermore, a given SIMPL process has no way of discovering what language was used to construct another IC with which it is exchanging a message.
- A SIMPL software IC has no way of discovering or knowing the physical location of its exchange partner. This means that the same binary image can be tested with local message exchanges and then deployed with remote message exchanges. Overall collective application performance would differ but the individual software IC would not need to change in any way. In may instances even a recompile is unnecessary.
- A SIMPL software IC has no way of discovering the internal algorithm of an exchange partner. This means that test stubs can be created which completely simulate and replicate the environment in which a given software IC finds itself. In particular, error conditions which would be costly or difficult to reproduce in the full system can readily be simulated in a test environment. These SIMPL software ICs can be rigorously tested before being deployed in the real-world application.

- SIMPL software ICs lend themselves well to projects with multiple developers. The implementation details of a SIMPL enabled process cannot affect any interacting process, provided that implementation conforms to the agreed upon message API. While a poor IC implementation will certainly adversely affect the overall application performance, the application will still operate. Once a poor algorithm has been identified it can be worked on in isolation from the real application and substituted once tested without even recompiling the adjacent ICs.

Basic Building Blocks

The most basic SIMPL processes types are:

- senders—those processes that compose messages and wait for replies
- receivers—those processes that wait for messages and compose replies

Listing 1

Listing 2

All the software ICs discussed in this section will be composites or special types of these two basic building elements. The SIMPL project is governed by the LGPL or similar open licenses. All of the source code for the software IC discussed below is available on the SIMPL web site.

Various Software ICs

Simulator One of the great advantages of the SIMPL paradigm is that it allows for the ready development of testing stubs. We have adopted the following naming convention with respect to these testing stub processes.

- the stub for a receiver process is called a “simulator”
- the stub for a sender process is called a “stimulator” A typical simulator setup might look as follows:

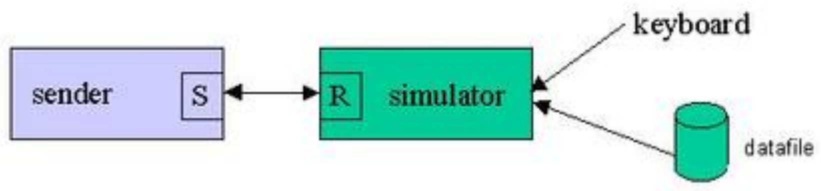


Figure 2. Simulator

The item being tested here is the sender. The key to understanding simulators is understanding the fact that, provided the simulator conforms to the SIMPL naming convention expected at the sender and conforms to all the expected message formats that the sender can exchange, the sender will not be able to detect that it is talking to a test stub. As such, the sender process can be vigorously tested in a very realistic test “sandbox” without having to alter the deployed executable in any fashion. There is no need for conditional compiles, test flags, etc., that are typical of unit test scenarios in non-SIMPL designs. Once tested, the sender executable can be deployed as is in the final application.

The exact composition of the simulator code is highly dependent on the application. The diagram above illustrates a typical scenario whereby one desires to have the ability to interact with the simulator directly via keyboard commands. In addition the canned responses are being fed in from a data file.

One can imagine more sophisticated simulators where the whole test sequence is metered in from the data file in a highly controllable manner.

Stimulator

When the object needing unit testing is a receiver process, one would typically use a stimulator to replace the real senders in the test phase.

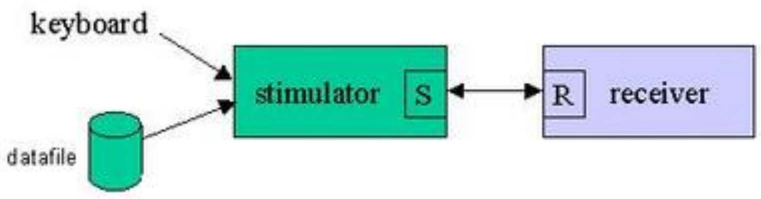


Figure 3. Stimulator

The item being tested here is the receiver. As was the case with the simulators above, the key here is that, provided the stimulator conforms to all messaging and naming conventions, the receiver process will have no way of knowing that it is being sent a message from a stimulator or from the real sender in the final application.

As was the case with the sender process in the simulator example, the receiver under test here can be the final deployable executable in all respects. Once again no conditional compilation or other executable altering techniques are required in the SIMPL paradigm.

As with the simulator, the typical stimulator contains a keyboard interface for the tester to interact with. More sophisticated stimulators may feed the test input from a data file.

The importance of being able to test deployable executables in a SIMPL application cannot be emphasized enough. In our experience this is one of the most important reasons for considering the SIMPL paradigm in designing software applications.

Relay

In a SIMPL system all processes are named. In the simplest of systems the names are normally assigned to processes (and passed to other processes) as part of the startup information on the command line. Occasionally it is desirable to simplify the amount of name information that needs to be passed into the code. The construct called a relay is useful for this type of thing.

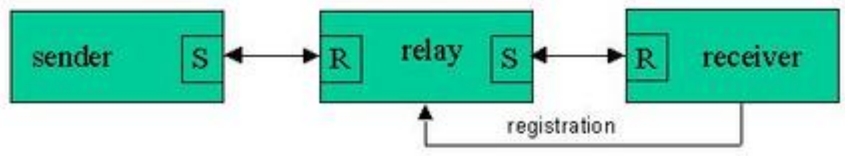


Figure 4. Relay

The basic relay operation is quite easy to grasp. The sender thinks that the relay process is the intended receiver for its messages. It does all the normal name locate and send operations as if it were part of a simple sender/receiver pair. The relay process on the other hand does nothing at all with the message. It simply remembers the ID of the sender and then forwards that ID on to the registered receiver process. When the receiver gets the relayed message it connects to the shared memory block of the sender process and retrieves the message in the normal manner. Once the message is processed the receiver places the reply in the sender's reply area and replies the sender ID back to the relay process. The relay process then simply replies back to the sender in the normal manner. Note that because of the SIMPL architecture, the relay process never needs to copy the message during these activities.

The advantages of this construct over a basic sender/receiver pairing lie in the name hiding that can occur with the receiver. It is also possible to start and stop receivers dynamically in this scheme without having to recommunicate naming information to the various senders in the system. If that occurs the startup message exchange, called registration in the diagram above, takes care of notifying the relay task of the new receiver's name information.

The ability to start and stop processes dynamically without cycling the whole application can be a significant advantage, particularly if the receiver logic is undergoing frequent upgrades or enhancements. These can be rolled in dynamically, and if problems occur the original copy can quickly be rolled back into play. In fact, with the registration scheme, both receiver processes could be running and a quick message exchange will have the effect of “routing” messages to the new receiver (or back again). With the registration scheme, the receiver in question can override an existing registration. While some may view this type of thing as a potential security hole it is only open to someone with privileges for running a new process on the system. It is relatively easy to build a certificate-type check on top of the registration process to close this hole considerably if that is an issue.

Obviously the relay will incur a performance penalty over the straight message exchange, but in many circumstances the advantages which come with the construct outweigh the disadvantages. The relay is a powerful SIMPL construct.

Agency

In addition to the name-hiding capabilities of the relay construct, it is sometimes desirable to exert greater control on the messages and their sequencing. The agency is a useful construct for these types of applications.

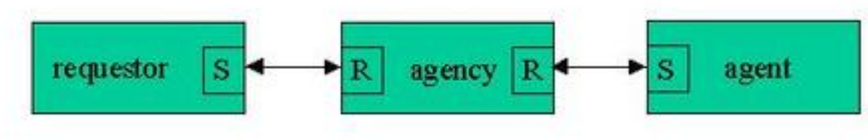


Figure 5. Agency

To understand the agency, one needs to understand the concept of reply blocking. In a normal SIMPL message exchange the receiver is receive blocked. Once the sender sends a message it is then said to be reply blocked. The key to the agency construct is that the receiver does not need to reply right away to that particular sender. It can simply remember the ID and go on about its business. In fact the receiver can “hold” the sender waiting and go back to being receive blocked for a new message. When new information arrives via a message from a second sender the receiver could choose to reply to the original sender with that information using its previously remembered ID. Another way to look at a reply blocked sender is as a “receiver” that doesn't block its “sender”.

To avoid some confusion of semantics, we have adopted the naming convention for the agency processes as per the diagram above. The requestor is simply another name for the normal sender. As far as this process is concerned the intended receiver for the message is the agency itself. The agency process however is completely neutral to the actual message content. It is simply going to act as “store and forward” for the requestor(s) messages. It is important to note that with the basic SIMPL package all “sender” type processes place their message in a block of shared memory which they own and control.

The actual message does not need to be copied out of the sender's buffer by the receiver but can be read directly by linking to the shared memory area. The agency construct takes advantage of this fact. When the requestor sends a message to the agency, the agency does not copy the message anywhere. It simply notes the ID of the requestor and does one of two things:

- Queues the requestor ID
- Forwards the requestor ID on to the agent process

The agent in this scheme contains all the normal “receiver” logic for this application. It is, however, working as a reply blocked sender. In its simplest form the agent talks to the agency in three different message formats:

- **WHAT_YA_GOT** request for any queued messages
- To which the agency will reply **GOT_ONE** requestor messages for it to process
- To which the agent will respond by sending **AGENT_REPLY** messages which contain the processed reply destined for the requestor

It all looks quite complex, but in fact it is quite SIMPL. Imagine the following scenario. The agency and agent are up and running. The agent locates the agency and then says WHAT_YA_GOT. At this point, because the requestor has not sent in any message the agency simply notes the agent's ID and leaves him reply blocked. At this point suppose the requestor generates a message and fires it off to the agency. The agency receives the message and notes that the agent is ready to process it. It simply relays the requestor's ID on to the agent via a SIMPL reply and leaves the requestor reply blocked. The agent then is free to chew away on the requestor's message while the requestor is reply blocked in the normal manner. Ultimately the agent will come up with a “response” to the requestor's message. It wraps this response in its AGENT_REPLY message and fires it down to the agency. Part of this wrapper message contains the ID for the requestor process. The agency then unwraps the message and replies it back to the requestor and leaves the agent reply blocked waiting for the next request.

To the requestor it all went exactly as if it had been sending any SIMPL message to a basic receiver. In fact there is no difference in the requestor code for dealing with agencies. Why then go to all this trouble?

First of all, it is now possible to dynamically start and stop the agent process in this system without affecting the requestor (other than delaying responses to a request that arrived while the agent was being cycled). In systems where the agent might be undergoing significant revisions or upgrades, this might be a distinct advantage.

Secondly, the requestor in this system does not need to know the name of the agent in order to exchange a message with it. The agency construct can be viewed as a message gateway.

To understand the further advantages we need to examine the case where we may have multiple requestors all talking to the same agency and agent. In this scenario the agency will actually receive all the requestor's messages and will queue their originator's ID's. The agency logic can then be in control of the order in which these messages are dispatched to the agent. In a normal sender/receiver pairing the FIFO imposes a first-in, first-out ordering and it is not possible to have a higher priority message jump ahead in the queue. In the agency scheme this is very possible.

In addition, in the normal SIMPL sender/receiver pairing the messaging is synchronous. It is intentionally difficult to kick a sender out of a reply blocked state by any other means than by having the receiver do a reply. This means things like timeouts or “aged data” are difficult to handle. The agency scheme makes these things relatively easy to manage. While messages are pending in the agency queue, the agency can be kicked into examining these periodically for timeouts or aging.

The agency construct will suffer a performance penalty when compared with a basic sender/receiver pair because at least two extra messages need to be exchanged in each transaction. The agency construct, however, is a powerful one and can be used to great advantage in certain designs.

Courier

Occasionally it is necessary in a design for two receiver processes to exchange messages. The courier construct illustrated below is a good way to accommodate this requirement.

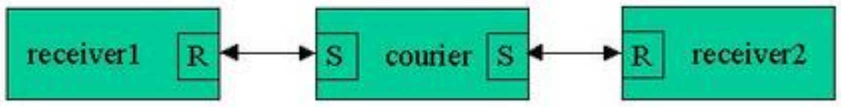


Figure 6. Courier

A typical example would involve a user-interface process. Typically user interfaces, be they simple text-based interaction or GUIs, want to be receiver-type processes. It is not often that you would want the user interface to block on a send. Very often in these designs the user interface (UI) requires information from another receiver process. If you went ahead and coded a blocking send into the UI then you could potentially have a place in the operation of the UI where the interface would freeze while the request was being serviced. This may not be the desired behavior.

The courier construct takes advantage of the delayed reply concept illustrated in the agency construct above. In our discussion we will assume that the UI process is “receiver1” and the recipient process is “receiver2”. When the courier process is started the first thing it does is locate the UI process it is designated to service. Once located, the courier will send a registration-type message to that process indicating that it is ready for action. The UI process will simply note that the courier is available and not reply, thereby leaving the courier reply blocked. At the point in the UI where the asynchronous request to the receiver2 process needs to be accomplished, a message is composed and sent (replied) via the courier. The courier is now unblocked and proceeds to locate and forward the message to the receiver2 process using a blocking send. At this point the courier is reply blocked on receiver2 and the UI is completely free to do other things as permitted by its logic. When receiver2 replies to the courier, the courier simply forwards that reply on to the UI process using a blocking send and once again becomes reply blocked on the UI. The UI receives this message in the normal manner, notes that it came via the courier, marks that the courier is once again available and processes the message in accordance with the logic coded.

This simple courier described above is a single request version. If a second UI request intended for the receiver2 process is generated within the UI before the courier returns its first response that request will be refused, siting the “busy courier”. A simple enhancement to this single request logic is to have a single-message-queuing capability in the UI. The “busy courier” response then would only come if a third UI request is attempted before the original response is received. In most UI processes this single message queue is more than adequate. A larger queue depth algorithm could be constructed readily, but the need for this is often indicative of a poor UI design elsewhere.

Another variation on the courier model is to have a parent process fork the couriers on demand. In some cases this capability is more desirable than having the courier prestarted along with the GUI process. The web applet type GUI applications are examples where this courier spawning technique is desirable.

Especially in user-interface designs, the courier construct is a very useful SIMPL building block indeed.

Broadcaster

There are times in a design where there is a need for a one-to-many sender/receiver relationship. For simple cases, one can simply have the sender locate all the intended recipients and loop through sending to each. In more sophisticated designs the broadcaster construct illustrated below is very powerful.

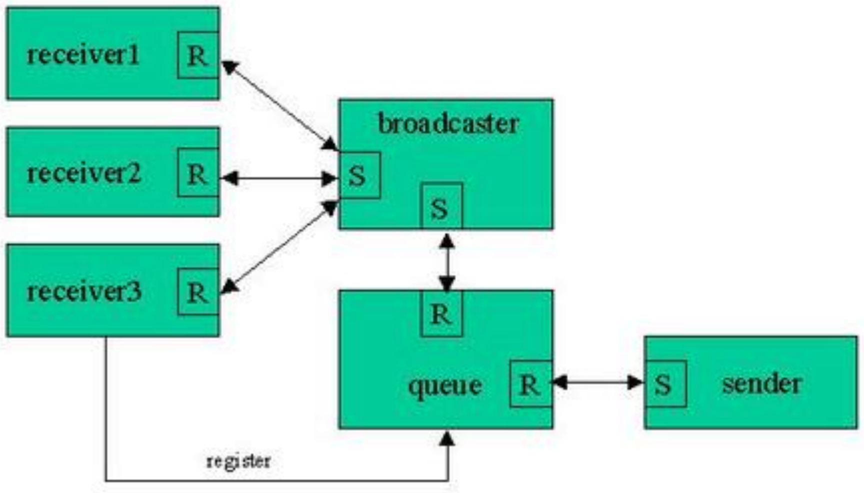


Figure 7. Broadcaster

The broadcaster actually consists of two parts: a receiver and a sender. We call the sender part the broadcaster. The receiver is typically a message queue as we shall see shortly. It works in the following manner: and the queue looks after message queuing and sequencing. The broadcaster maintains a list of processes to send to.

A typical sequence may start with a receiver (say receiver1) deciding that it wishes to receive broadcast messages. As part of that sequence it sends a registration-type message to the broadcaster's queue process. The queue will then place a REGISTRATION-type message onto its internal queue. Meanwhile the broadcaster returns from one of its broadcast sequences by sending a message down to the queue process asking whether there are any new messages queued. In this example the REGISTRATION message for receiver1 is delivered as a reply to the broadcaster. When the broadcaster process detects that the message is a new REGISTRATION, it does a nameLocate on that the recipient (receiver1 in this example) and stores the ID in its internal broadcast list. It sends a confirmation message back to the queue process, which then proceeds to reply and unblock the original receiver (receiver1—who was temporarily a sender). If there were no more messages on the internal queue, the broadcaster would simply be left reply blocked at this stage. At this point the sender may send a message to the broadcaster's queue process that is intended for broadcast. Typically the queue would queue the message and reply immediately to the sender, but one could do a blocking send scheme similar to that of the registration process. If the queue detects that the broadcaster is reply blocked it immediately forwards the message via a reply to the broadcaster. Once the broadcaster gets the message it notes that this is not a registration and therefore is a message to be sent to all the registered recipients in its broadcast list. Once this series of sends is complete the broadcaster will send back to the queue for the next message and the process repeats.

When a recipient wishes to cancel its registration with the broadcaster, it simply repeats the registration process with a DEREGISTER message to the queue. It is typical that the queue would simply queue and acknowledge this request.

If a recipient “forgets” to deregister and simply vanishes, the next broadcast attempt will detect that condition and the broadcaster would proceed to remove that ID from its internal broadcast list.

The broadcaster construct is a very powerful SIMPL tool. A typical example of its use would be to synchronize multiple instances of a GUI applet with the same information.

Conclusion

The SIMPL paradigm adds some important tools to the Linux developer's toolset. With its process centric model of encapsulation, coupled with a blocking Send/Receive/Reply messaging, the SIMPL libraries make an excellent platform on which to develop software ICs. We believe that the advantages of this software development paradigm are significant and cost effective.

All of the source code for these Software ICs is available on the SIMPL web site. While this source code delivery method for these ICs is effective as a means to “seed the thinking” it does not mean that SIMPL ICs must be delivered in source code format. There is nothing in the LGPL license that the SIMPL project uses that prevents software ICs from being deliverable in binary format.

This opens up an exciting era is software design. Software designers may finally be on par with their hardware cousins when it comes to managing project complexity.

Resources

Robert D. Findlay (fcsoft@attcanada.ca) has been involved in software development for over 20 years. While many of the projects over the past 15 years have involved QNX and various UNIX systems, the past two years have been exclusively LINUX. In his endless quest for “there must be a simpler way” he co-founded FCsoftware five years ago. When not working to keep the business afloat, he enjoys spending time with his wife Gloria and their two large dogs at their country home—built in 1860.

Archive Index Issue Table of Contents

Advanced search

Gnu Queue: Linux Clustering Made Easy

W. G. Krebs

Issue #79, November 2000

Farm those jobs out with Gnu queue!

So, your organization has finally decided to double the number of Linux workstations in your cluster. Now you've got twice as much computer power as before, right?

Wrong. It's not that simple. Old habits die hard, and your organization will probably continue trying to submit most of its jobs to the old computers. Or, used to the old computers being overloaded, your users will submit most of their jobs to the new computers, leaving the old ones idle. Let's face it, it's just too much of a pain to log into every computer on your network to see which one's the least utilized. It's simpler just to send the job *somewhere* and get on with the rest of the day's work, especially if it's a quick and dirty job and there are lots of computers. The result, however, is slower overall performance and wasted resources.

What you need is a simple utility for sending your job to the least utilized machine automatically. You could install a batch processing system like NQS—maybe you've already installed one—but it's annoying to check your e-mail or run special commands to see if your quick and dirty job has finished running in some batch queue. If something goes wrong, you might need to use nonstandard commands or track down which remote machine is executing your job, do a **ps** to learn its process id, and then do a kill. Users moving to new departments or new jobs often find that they need to relearn a complex set of nonstandard commands, because their new organization uses a different batch processing system than what they're used to.

You'd like something really simple, something that works through the shell, so that you could check your job's status with a command like **jobs**, and allow the shell to notify you when the job has terminated, just as if you were running it in the background on your local machine. You'd like to be able to send the job into the background and foreground with **bg** and **fg** and kill the job with **kill**, just as if the job were running on the local node. This way, you can control remote jobs using the same standard shell commands you and your users already know how to use.

Enter GNU Queue. GNU Queue makes it easy to cluster Linux workstations. If you already know how to control jobs running on your local machine, you already know how to control remote jobs using GNU Queue. You don't even need special privileges to install and run GNU Queue on your cluster—anyone can do it. Once you've discovered how incredibly easy it is to cluster Linux environments with GNU Queue, you'll wonder why organizations continue to spend so much money on comparatively hard-to-cluster Windows NT environments.

Quickly Configuring Heavily-Used Software to Farm out Every Time

With GNU Queue, all you have to do is write a simple wrapper shell script to cause software applications to farm out every time to the network:

```
#!/bin/sh<\n>
exec queue -i -w -p
-- realbogobasicinterpreter $*
```

and name it “bogobasicinterpreter”, with the real bogobasicinterpreter renamed “realbogobasicinterpreter”. This assumes, of course, that you have administrative privileges for your cluster (not necessary to install and run GNU Queue). When someone runs bogobasicinterpreter, GNU Queue is told to farm the job out of the network.

Another popular way to use GNU Queue is to set up an alias. You can do this even if you don't have administrative privileges on your cluster. If you are using **csh**, change to your home directory and add the following line to your .cshrc:

```
alias q queue -i -w -p --
```

and run the command **source .cshrc**. Then, you can simply farm out jobs by typing “q” before the name of the job you want to farm out.

Either way, GNU Queue does all the hard work, instantly finding a lightly loaded machine to run the job on. It then fires up a proxy job on your local machine that “pretends” to be the remotely executing job, so that you can background, foreground and kill the remotely running job through normal shell commands. There's no need to teach other users new commands to interact with some complicated batch processing system—if they understand how to use the UNIX shell to control local jobs, they understand how to use GNU Queue to control remotely executing jobs.

Advanced Features

Of course, GNU Queue supports many additional features. It supports a traditional batch processing mode, where output can optionally be returned by e-mail. Versions 1.20.1 and higher now have alpha support for various modes of job migration, which lets the administrator to allow running jobs to actually move from one machine to another in order to maintain a constant load throughout the cluster. More importantly, GNU Queue allows administrators to place limits on the number of a type of job that can run (say, allow no more than five bogobasicinterpreter jobs to run on any node) or to prevent certain jobs from running when a machines's load is too great. For example, the bogobasicinterpreter can't be started if the load average exceeds five; running interpreters are suspended if the load average on the node exceeds seven. It's also possible to place restrictions on the time of day certain jobs may be run (no bogobasicinterpreters on Saturdays) or to have it periodically check the return value of a custom script to determine whether or not a program can be run. But, you'll probably never need these advanced features.

Obtaining and Installing GNU Queue: A Quick Look

All of this sounds great, you say. How do I obtain and install GNU Queue? You can download the latest release of GNU Queue from its web site at http://www.gnuqueue.org/. It's a participating project on SourceForge, and you can find all sorts of discussion forums, support forums and bug-tracking databases. Download the program from the web site, unpack it and then run:

```
./configure<\n>
make install
```

from the top-level directory. Run **make install** and fire up the dæmon with **queued -D &** on each machine in your cluster.

For a quick reference on using the **queue** command to farm jobs out to the network, visit the GNU Queue home page. That's all there is to it!

Detailed Instructions on Installing GNU Queue

Before installing GNU Queue on your cluster, you have to make a decision that is basically guided by whether you have root (administrative) privileges on your cluster. If you do, you'll probably want to install GNU Queue in a manner that makes it available to all the users on your site. This is the **--enable-root** option. On the other hand, if you're just an ordinary Jane or Joe on your cluster or want to see what the fuss is all about without giving away privileges, you can install GNU Queue as an ordinary user, the default mode of installation.

Yes, ordinary users can install GNU Queue as a batch processing system on your cluster! But, if another user wants to run GNU Queue, he'll have to change the port numbers in the source code to insure no one else is running GNU Queue. That's why it's better to let the system administrator install GNU Queue (with **--enable-root** option to the configure script) if you expect a lot of users will want to run GNU Queue on your cluster.

Once you've downloaded GNU Queue off the Net, the first thing to do is to unpack it using the tar command. Under Linux, this is just **tar xzf filename**, where file name is the name of the file (compressed with **gzip** and having either the .tar.gz or .tgz file extensions. On other systems it's a little bit more involved, since the tar installed by default is not GNU tar and doesn't support the **z**decompression option. You'll need to explicitly run the **gunzip** decompression program: **gunzip filename.tar.gz**; **tar xf filename.tar**, where filename.tar.gz is the file, with .tar.gz extension, that you obtained from the network. (Savvy users might want to use the **zcat filename.tar.gz|tar tf -** trick, but this assumes the **zcat** program installed on your system can handle GNU zipped file. **gunzip** is part of the GNU **gzip** package; you can obtain it from ftp://ftp.gnu.org/.

So you've unpacked the distribution and you're sitting in the distribution's top-level directory. Now what? Well, if you're an ordinary Jane or Joe you install the program into the distribution directory by running **./configure** followed by **make install** on each machine in your cluster. Then, fire up the dæmon with **queued -D &** on each machine in your cluster. If you want more details (or you're a system administrator), continue reading.

Installation by Plain Folks

Run ./configure. If you're installing it on a system where you're not a superuser but an ordinary peon, configure sets the makefile to install GNU Queue into the current directory. **queue** will go into ./bin; queued dæmon will go into ./sbin; ./com/queue will be the shared spool directory; the host access control list file will go into ./share; and the queued pid files will go into ./var . If you want things to go somewhere else, run **./configure --prefix=dir**, where **dir** is the top-level directory where you want things to be installed.

System-Wide Installation by Superusers

The default ./configure option is to install GNU Queue in the local directory for use by a single user only. System administrators should run the command **./configure --enable-root** instead. When installing with the **--enable-root** option, configure sets the makefile to install GNU Queue under the /usr/local prefix. **queue** will go in /usr/local/bin; queued dæmon will go into /usr/local/sbin; /usr/local/com/queue will be the shared spool directory; the host access control list file will go into /usr/local/share; and the queued pid files will go into /usr/local/var. If you want things to go somewhere else, run the following:

```
./configure --enable-root<\n>
--prefix=dir
```

where **dir** is the top-level directory where you want things to be installed.

./configure takes a number of additional options that you may wish to be aware of, including options for changing the paths of the various directories. **./configure --help** gives a full listing of them. Here are a few examples, **--bindir** specifies where queue goes; **--sbindir** specifies where queued goes; **--localstatedir** states where the spool directory and queued pid files go; and **--datadir** lists where the host access control file goes. If ./configure fails inelegantly, make sure **lex** is installed. GNU flex is an implementation of lex available from the FSF, http://www.gnu.org/.

Now, run make to compile the programs. If your make complains about a syntax error in the Makefile, you'll need to run GNU Make which is hopefully already installed on your machine (perhaps as **gmake** or **gnumake**), but, if not, you can obtain it from the FSF at http://www.gnu.org/.

If all goes well, make install will install the programs into the directory you specified with ./configure. Missing directories will be created. The host name of the node make install is being run on will be added to the host access control list if it is not already there.

Now, try running Queue. Start up **./queued -D &** on the local machine. (If you did a make install on the node, the host name should already be in the host access control list file.)

Examples and Options

Here are some simple examples:

```
> queue -i -w -n -- hostname<\n>
> queue -i -r -n -- hostname
```

For a more sophisticated example, try suspending and resuming it with **Control-Z** and **fg**:

```
> queue -i -w -p -- emacs -nw
```

If this example works on the localhost, you will want to add additional hosts to the host access control list in share (or --datadir) and start up queued on these.

This line:

```
> queue -i -w -p -h hostname -- emacs
-nw
```

will run Emacs on host name. Without the **-h** argument, it will run the job on the best or least-loaded host in the Access Control List file. There is also a **-H hostname** option, which causes hostname to be preferred, but the job will run on other hosts if hostname is unavailable.

At this point, you might be wondering what some of the other options for queue do. **./queue --help** gives a list of options to Queue. The "--" separates GNU Queue options from the options to be given to the command to be run. **-i** stands for immediate; it places the job to be run in the "now" batch queue. **-w** invokes the proxy job system, as opposed to **-r**, which causes output to be returned to the user via e-mail (traditional batch processing mode). **-n** turns off virtual terminal support. Most users will probably only use **-i -w -p** (full virtual terminal support, for interactive jobs like Emacs) and **-i** w -n (no virtual terminal support, for noninteractive jobs).

More details on the protocol GNU Queue uses for host selection can be found in the on-line manual and the on-line Internet draft protocol at http://www.gnuqueue.org/.

Segregating Jobs Using Spool Directories

You can also create additional queues for use with the **-q** and **-d spooldir** options. They might be used to specify different queuing behavior for different classes of jobs. Each **spooldir** must have a profile associated with it. The profile determines queuing behavior for jobs running in that spooldir. See the on-line manual for more details.

Fine-Tuning Cluster Performance

That's all there is to it! Of course, for GNU Queue to work well there needs to be some sort of file sharing between nodes in the cluster (for example, NFS, the Network File System). If you have the same home directory, regardless of which machine you log into, your system administrator has somehow configured your home directory to be shared across all cluster nodes. You want to make sure that enough of the file system is shared (i.e., is the same) between cluster nodes so that your programs don't get confused when they run. Typically, you'll want system temporary directories (/tmp and /usr/tmp) to be non-shared, but everything else (except maybe the root file system containing kernel images and basic commands) to be shared. Because this configuration is so common to UNIX and Linux clusters, we've assumed here that this is the case, but it isn't necessarily so; so check with your system administrator if you have questions about how files are shared across your network cluster.

Documentation and Mailing Lists

Documentation about GNU Queue is also available off the web site, including an Internet draft on the protocol GNU Queue uses to farm out jobs. While you're there, you'll probably want to sign up for one of the three mailing lists (queue-announce, queue-developers and queue-support) so that you can learn of new features as they're announced and interact with other GNU Queue users. At the time of writing, queue-developers is by far the most active list, with lively discussion of improvements to GNU Queue's many features and suggested ports to new platforms. You can obtain advice for any problems you encounter from the queue-support mailing list.

CVS Repository: Joining the Developer Community

Another SourceForge feature mentioned on the home page is the CVS repository for GNU Queue. Interested readers can obtain the latest prerelease development code, containing the latest features (and bugs) as they are added by developers, by unpacking the GNU Queue distribution and running the command **cvs update** inside the top-level directory. If you're actively making changes to GNU Queue, you can apply for write access to the CVS directory and instantly publish your changes via the **cvs ci** command. If you can get other developers interested in your work (via the queue-developers mailing list, of course), you can bounce code changes back and forth amongst yourselves via repeated cycles of cvs, ci and cvs update. All of this assumes you have cvs installed, which is the default with many Linux distributions.

Code isn't the only way interested readers can contribute to GNU Queue. There are many ways to contribute to the GNU Queue effort on SourceForge. With a login on SourceForge, one of the project administrators can give you editor privileges for the documentation tree, moderator privileges in the discussion forums, or administrative privileges in the bug tracking and patch database sections of the site.

Getting Help

If you encounter problems with installation not explained here, you may wish to check out the support forum and support mailing list, available off GNU Queue's home page, http://www.gnuqueue.org/. Bugs should be reported to bug-queue@gnu.org.

Farm out that Job!

So remember: the next you have a quick and dirty job to run, don't waste time or resources. Farm that sucker out using GNU Queue!

W. G. Krebs is a PhD candidate in molecular biophysics and biochemistry at Yale University, where he researches web-based biological databases. He has been a systems programmer for longer, and in more languages, than he cares to relate. His wide-ranging interests include political economics, classical and folk music, and the Chinese game of Go; he welcomes your comments at wkrebs@gnu.org or by snail mail c/o *Linux Journal*.

Customize Linux from the Bottom—Building Your Own Linux Base System

He Zhu

Issue #79, November 2000

Can't find a system that has everything you want? Build your own.

We have seen various Linux distributions, and yet many others continue to appear. Some are as small as DLX which sits on a single floppy; others are as big as Red Hat 6.2, packed in five CDs. Things seem to become more complex and harder to manage as systems grow. How is a Linux system put together from pieces of free code? How can we assemble and customize our own system for a particular purpose? It seems to be a hard task.

However, from the view of the base system, in principle, all distributions are assembled in a similar way. The difference is that big ones are armed with more packages and more fancy stuff targeted at more general audiences, and small ones have fewer goodies and are aimed at relatively narrow and specific user groups.

Fully featured Linux distributions are usually unnecessarily big for specialized situations. For example, embedded applications need a slim base for their particular situations, and there are already small Linux distributions available for these purposes. However, because there are so many factors to consider, no one can claim that their distribution is comprehensive and satisfies all customers.

Usually an application needs a customized base system to work efficiently. You can pay for a solution from many Linux service providers, or you can do it yourself. Sometimes knowing how to build the base system on your own is more beneficial. With such skills and knowledge, engineers can easily control and improve the system's behavior for the needs of their customers. DIY (Do-It-Yourself) is not just fun, but strategically important in some cases, and more people are recognizing the value Linux offers. By doing so, you acquire the ability to customize not only the Linux kernel but also all other components of your system to achieve the optimizations best suited for your requirements.

This article tries to tell readers that building your own base Linux is not a daunting task. It tells our experiences and gives a brief introduction to building and customizing a Linux system. It is a base system: small, clean and ready to go. We try to make the complex simple, without losing generality and effectiveness. We show how to make the building steps as easy as 1-2-3, and how to customize this system to be minimal, accommodated in one floppy. After all, it should be good enough to be used as a start point for a base system to run any typical applications. This is possible because we build the system directly from unchanged sources. This allows us to always use the latest stable versions and make all required kernel services available.

Linux System

A system is simply defined as a combination of inherently connected parts. A Linux system (only considering software in this article) is a combination of a Linux kernel and other components which make the kernel useful. All software components except the kernel need to be resident in a root file system (there may be a few other file systems, but they must be mounted under the root tree to become visible). So, technically, we can simply consider a Linux system as a combination of a kernel and a root file system. All Linux distributions are arranged in this way. For example, a fully installed Linux system is a kernel plus a big root file system. A Linux installation disk and a rescue system are used to install a full system and to repair problem systems, respectively. They are also organized in the same way; that is, consisting of a kernel and an initial root file system, but the initial root file system is small and only holds a few basic components necessary to do limited jobs (see Figure 1). The Linux kernel has been coded to take special measures to locate the root file system, either from a normal file system or from a compressed image of an initial root file system.

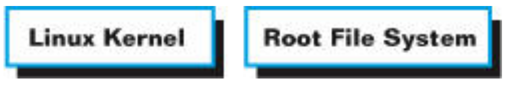
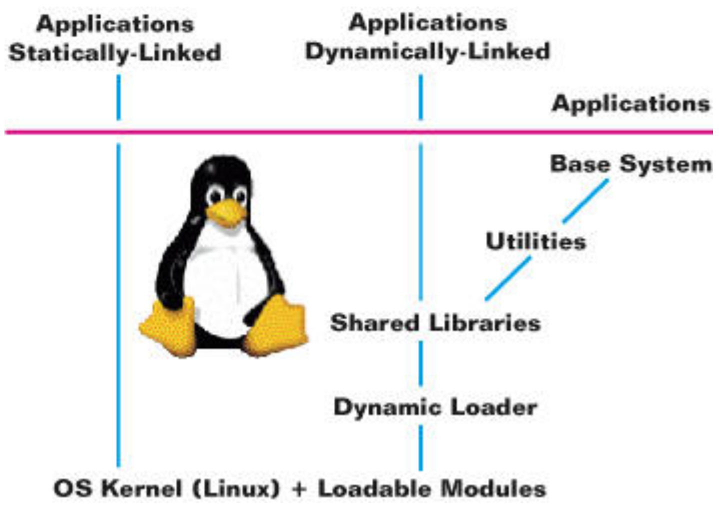


Figure 1. A Linux System

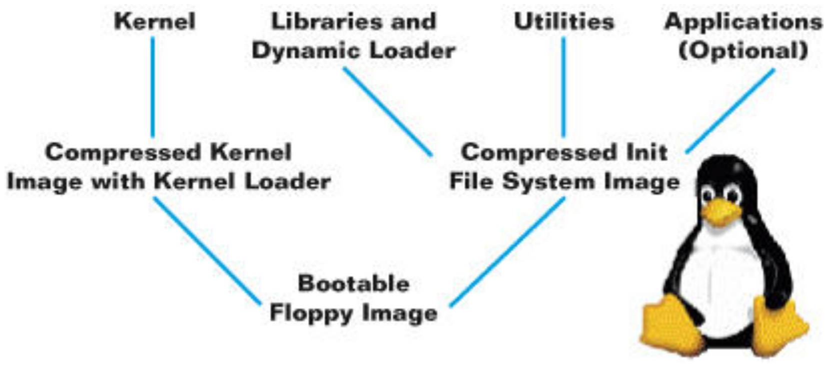
The Approach

Given an application, we want to run it above the kernel using shared libraries on a box. Assume the application doesn't require any rare features which Linux doesn't provide at the moment. We want a base system that can run this application and provide some basic control and management as well. A typical case of the base system is shown in Figure 2. Note that this figure is not complete because a utility may be statically linked, which doesn't require a shared library and a utility may be an a.out style, which doesn't use a dynamic loader.


Applications Statically-Linked
Applications Dynamically-Linked
Applications
Base System
Utilities
Shared Libraries
Dynamic Loader
OS Kernel (Linux) + Loadable Modules
Figure 2. A Base System

If the application is self-sufficient, that is, statically linked with everything required at runtime, it can run right over the kernel without any support from shared libraries. The base system in this case may mean only the Linux kernel itself. However, almost all systems need support from one or more utilities to manage things like file operations and system monitoring, using commands like mount and ps. We consider a base system to be a combination of the kernel, the dynamic loader, a set of libraries and a set of utilities.

Like many other systems, our goal is to show how to create the system on a floppy with both the kernel and a compressed image of an initial root file system, as shown in Figure 3. This compressed initial root file system will be uncompressed by the kernel and put into a ramdisk, that is, a space of RAM set aside to hold a small Linux file system. Creating such a base system is often straight-forward, but rather tedious for most people. We simplify the procedure. The idea is to design a well-organized hierarchy of makefiles which will extract sources, compile, and setup the contents of the initial ramdisk, then prepare and pack the whole system.


Kernel
Libraries and Dynamic Loader
Utilities
Applications (Optional)
Compressed Kernel Image with Kernel Loader
Compressed Init File System Image
Bootable Floppy Image
Figure 3. Arrangement of System Components

Customization of the above base system depends on the requirements of the application that will run on the box. Choose a configuration for the kernel, a dynamic loader, a set of libraries that are necessary for the application and utilities, and a set of basic utilities which are required to control and manage the system. Then, compile all these things in a consistent environment. After that, package the results and make it bootable. If something is missing, you are free to add it to the list, and make it again.

Our Modest Goal

A curious reader might ask why we do this and what it is good for. Like many others, we want to run some complex software on a box, something that can be generalized as a multitasking application on a typical PC-like machine without hard disk or monitor. We need an OS kernel and some elements as the base experimental platform. This platform should be robust, maintainable and customizable. Writing a good OS kernel for this purpose is too scaring for many. Thanks to Linux and the Open Source community, we now have an excellent option.

Basic materials are ready and available for free. Now it is time to pick up pieces we need, assemble our own engine and control it. Then, it is time to enjoy.

Before we start, we need to know the answer to some key questions: How to compile the kernel? How to compile a shared library? How to create an initial root file system? How to put the kernel image and compressed file system onto a floppy or EPROM? How to run an application using shared libraries? How to debug? There are many questions like these. The answers are already documented, not, as far as we know, in a single place, but scattered over a wide range of documents. We don't want to write a comprehensive document for these questions but, rather, tell our story and major part of our answers.

Steps

Once the plan has been made for customization, detailed steps can be put into action. General steps in our work are described in the following.

1. Setup the development environment Install a full Linux distribution such as Red Hat 6.0 as the development platform. Make sure it's gcc-supported. To make things easy, assume the target machine on which we run our customized system and the host machine, that is, the development machine, are using the same type of CPUs, in our case Intel x86. Otherwise, we have to prepare a cross-compiler. 2. Customize the kernel Get the latest stable kernel source (version 2.2.13 at this article's writing). How to configure and compile the kernel are well documented in the source files. We don't want to repeat the details here. But we need to choose support for initial ramdisk, loadable modules and

other necessary options. If we use the serial console port, choose serial console support. If we have a piece of hardware we want to set up, like an Ethernet card, we can select them as modules. Then we can install and use these modules in our base system. 3. Prepare the standard libraries Get the latest stable standard library glibc (versions 2.2.1). This includes almost everything we want: the dynamic loader, the standard C library and math lib, etc. Although we don't need all that glibc provides, it's better to build them all together, then choose what we want. The documents in the glibc source tree tell in detail how to compile it. Because we use it for our target machine, we have to compile it using the kernel header files consistent with Step 2. This can be specified using **--with-headers** during configuration. Also, we have to install all library header files so that the compilation of other components for the target machine can use them. 4. Let the compiler know how to cross-compile After installing kernel and glibc header files, we need to prepare the compiler, gcc, to use them. Glibc2-HOWTO describes this in more detail. Briefly, we need to tell gcc where to find specifications using the option **-b**. In our case, because the target and the host machine are basically the same, we need to use only the host machine specs. This can be found by using the command **gcc -v**. For example, on my box, the reply is:

```
Reading specs from
/usr/lib/gcc-lib/i386-redhat-linux/egcs-2.91.66/specs
gcc version egcs-2.91.66 19990314/Linux
(egcs-1.1.2 release)
```

Compile other components of the system by adding the option **-b** as:

```
gcc -b i386-redhat-linux
```

5. Select and prepare utilities and other libraries To compile a utility program such as mount or libraries other than those in glibc, such as termcap, consistently, we have to tell the compiler where to find all the header (include) files and the libraries. First, we tell the compiler not to search header files in default paths by specifying **--nostdinc**. Second, tell the compiler we are compiling sources for the target machine by using option **-b $MACHINE**, as described in Step 2. Third, state exactly where to find the kernel header files and the standard library header files by specifying option **-I**. Fourth, tell the loader what libraries to use and where to find them by using **-L** and **-l** options. 6. Build applications Like compiling for utilities and libraries, we need to cross-compile our applications in order to use them on the base system. There are no particular considerations needed from the system view, except to make sure everything on which the application is dependent is installed already. 7. Package things together Once all the components are ready, we need to arrange them in such a way that the system be booted, and any everything can be correctly located. This issue is discussed in more detail in the following section. 8. Move on Use the base system as a start point for whatever you want

to do. More features can be added, as needed, over the base system. A separate section is devoted to this issue.

Creating a Base System

As far as we know, it is hard to find a document that tell us in detail how to put images, executables, binaries and scripts together; in other words, to package things together, to assemble a system. Different systems may take different approaches to packaging, although components can be created in the same way. The easiest and most popular way is packaging on floppies. The general steps of packaging a bootable system on a single floppy, that is a boot/root floppy, can be summarized in the following few steps:

1. Create individual items needed for the system, such as the kernel image, libraries, executables, scripts and configuration etc.2. Create the directory structure of the initial root file system for the base system.3. Move things to the root file system and create items like device nodes.4. Create the compressed root file system.5. Tell the kernel where to find the initial root file system image by setting some flags in the kernel image.6. Write the kernel and the compressed root image to a floppy and make it bootable. To show more details, we wrote a set of makefiles. They are actually instructions to implement the above steps and to create a small base system from freely available packages. We run a simple application called a netperf sever which tests TCP/IP stack performance and is also freely available from the Web. We provide these instructions in Resources. It may not be simple to run, but curious readers can dig into the lines and find how to build a base system from scratch.

An application can be started in different ways depending on how the base system is configured. In most of cases, a Linux kernel is configured to run a startup script or a binary executable, called **init** or **linuxrc**, in the initial root file system after the kernel is up. This init program usually does things like remount the root file system to allow read/write permissions, mount other file systems like proc, and initialize other parts of the system, such as starting a shell interface or running the application immediately. The SysVInit program is very popular in most Linux distributions for this purpose.

For our base system, we don't need a complex init sequence to demonstrate. So, we simply write a shell script like the following. Anyone is free to change and add more commands to it:

```
mount -n -o remount,rw /
mount /proc /proc -t proc
echo  MyCompanyName, Version X.Y. Built Z, August 2000
exec /bin/sh
```

As an exercise, it might look better if you have the above echo line in your application, and start the application at the end of the script instead of running the standard shell. An example in C++:

```
cout << COMPANY << VERSION_NO << BUILD_NO <<
__DATE__ << __TIME__;
```

In our case, the system prompts after it is up.

```
pipe-elinux> MyCompanyName, Version X.Y,
Build Z, August 2000
pipe-elinux>
```

Making It More Attractive

After the base system is up, you might think it is not much use without any interesting applications. But it is a base from which you could start your big project. One by one, you can gradually add things into this base system, making it more and more attractive. The following examples might be worth considering:

- a init program : SysVInit is a good choice, but it seems too big for simple applications
- a security facility: add login support
- an editor : vi or emacs
- more networking services: telnet or ftp dæmons
- a GUI : X is a choice
- non-volatile storage: flash memory and hard disk support.
- add more loadable modules
- use rpm to manage packages

We don't really need to recompile every package we choose, because we can easily find a binary already compiled for a processor. Like our system, the host and the target machines are the same type, so, we can just use most of the binaries found on the host machine. For example, for the utility **top**, we just copy the binary into our base system; and then it runs. Things are not always that simple, however. Because most often we have to sort out the dependencies for an executable, that is, the shared libraries it needs and configuration files it reads, usually we don't know until we run it. However, some tools can help us with this. **ldd** and **strace** can tell us these dependencies. For example, once I tried successfully to run Emacs on the base system by simply copying the executable (emacs-nox), a few shared libs and configuration files to the base system. This usually helps us a lot during the development and saves a lot of time.

You might not be satisfied with booting from floppies. Instead, you can implement booting from EPROM or others. To do this, you have to redesign your packaging approach, but the components are mostly unchanged. What's specific here is the kernel image loader. Booting can be implemented like:

- boot from EPROM
- boot from Networks
- boot from devices like flash, hard disk partition, CD-ROM and ZIP disks; that is, any devices other than floppies
- boot from other OS

If you like and want to spend more time, you can make your system as fancy as many other successful systems already in the field.

Problem Solving

One of the advantages of using Linux is that there are many documents and tools to help you customize your system and solve problems. The code is no secret. Everything inside and outside is open. Besides, there are many other useful sources in print and on the Web. There is no other system which can compare to Linux in this respect, not even Free BSD, let alone any proprietary operating systems.

Typically, for a problem, we might work out a few different solutions. We always want to pick the best one, of course, but it is not easy to know which is the best until we have tried each of them. To solve a problem, in many cases, we can find answers by consulting Linux HOWTOS and docs, or asking Linux guys in our organization. As an alternative, you can post a message on a Linux newsgroup and hope someone on there can give you a quick reply. If you want to pay, there are many Linux-related companies providing technical services as well. (If the problem is stubborn, as the last straw, kick your buggy box a few times, as I did sometimes. You must be careful—don't break it and then reboot. That should work; otherwise repeat the problem-solving sequence from the beginning again.)

gdb is an excellent debugging support for applications on the base system. If we don't want or are unable to run a full gdb on the target system, that is, the base system, we can run small remote gdb facilities as either a gdb stub or a gdb server on the target. Besides, things like the syslogd dæmon can also help debugging on the target system.

There are many good problem-solving strategies. Whatever approaches we use, the goal is to find the proper solution. It is usually safe to follow a successful example. For example, we learn something by checking things inside a Red Hat rescue system. We can do this simply with the following few commands:

```
cat rescue.img | gzip -d > rescue_root.img
mkdir rescue_root
mount -o loop rescue_root.img rescue_root
```

Here rescue.img is the compressed rescue floppy image found in the Red Hat distribution's images directory. Then we can check its contents by:

```
ls rescue_root
```

It displays:

```
bin dev etc lib lost+found mnt proc sbin tmp usr
```

You get all the detail in the floppy.

Conclusion

This article is only an introduction to customization of the Linux base system. For a particular situation, it could be rather complex, especially when modifications at the code level are required, such as to support specialized hardware. But, we have shown that it is a manageable task. Our purpose is to make things simple in order to encourage people to take the challenge. By creating our own customized base system with a moderate effort, we get a power engine which can drive us into the bright future.

Resources

He Zhu (hezhu@yahoo.com) is interested in system software and networking. He is currently working for Bell Labs, New Jersey.

Linux as a Work Environment Desktop

Mark Stacey

Issue #79, November 2000

Tips and suggestions for using Linux on the desktop in a non-Linux workplace.

As we've seen over the last 18 months or so, Linux has stormed up the league tables in terms of server-side operating systems. Many people, in fact, are now beginning to see that there are options beyond a small number of large companies that traditionally held sway over the high-end operating system market.

The next logical step in this evolution of Linux is the migration from the back-end services, where it has proved itself admirably, to the desktop. This switch is more difficult to accomplish, as Linux was traditionally developed with server-side processing in mind. Until recently, there had been no concerted effort to develop applications for the end user that could compete against the dominant players.

While many argue that Linux is still not ready to act as a desktop OS, there isn't too far to go before this scenario becomes feasible. In this article, I aim to give you a feel for some of the challenges that Linux enthusiasts may meet when setting out to make this an environment more appealing to their sensibilities, as well as some tips on resolving them.

I work as a developer in a small office environment. The project that I'm working on at the moment has put me in an ideal situation to experiment with Linux as a serious desktop environment. I admit that this scenario may not be appropriate for everyone, but I attempt to deal with handling some of the more common tasks under Linux.

The project concerns Java development, with the main development platform being Solaris. The tasks that I've had to perform include designing the project, developing and testing code, tracking project defects, and general day-to-day administration, such as mail, research, and document reading and writing.

Design Tools

Most of the design work that takes part in the office involves the use of UML as the modeling language. In UML there are a number of conventions for representing different aspects of object-oriented program design. While there are a plethora of diagramming tools available for the Windows environment, tools for the Linux environment are generally lesser known, harder to come by, and maybe not as polished as their counterparts in the Microsoft world. Because the development environment that I work in is predominately Java, I had the advantage of not having to base my search wholly in the Linux application area; a good tool that is written in Java is cross platform by nature. While searching for suitable tools, I came across two likely candidates.

One of these tools is called ArgoUML. The web site for the tool, listed in Resources, describes ArgoUML as a cognitive design tool. It attempts to examine your design and provide suggestions on what may be lacking, or point out discrepancies in your design. The version of ArgoUML that was available from the web site was only a development version. They seem to have moved recently to a new site and revamped their development efforts.

On Windows, the Rational Rose UML tool has a pretty powerful presence in the object-oriented design world. Being able to read and write project files that are compatible with Rose is a great advantage. MagicDrawUML is a commercial UML design tool. Written in Java, it is completely cross-platform, and by exporting your projects in Rational Rose format, you can also share your work with others. As a side note, many people claim that Java can be quite slow, so what the developers did with this tool is pretty impressive. The user interface is as responsive as any you might use in a Windows or Linux environment. The downside of this product is that it doesn't come free. However, the license fees are a fraction of Rational Rose, and in a commercial development environment, it may be a worthwhile investment.

Development Tools

In this section, I try to give you some idea of the tools that are available for Linux that can be used to develop and build a Java project.

The build environment we use has been set up with **Make**. This tool is pretty standard on Linux installations. Make is available on a wide variety of platforms, so it is a good choice for cross-platform projects. The bulk of platform independent macros and targets can be defined in standard Makefiles. By using the following strategy, you can minimize the fuss that is involved in building over multiple platforms.

First, define an environment variable ARCH, which is set to whatever the **uname** command returns. This can be done at login time by adding the following command to your .bash_profile (for bash users):

```
export ARCH=<\#145>uname'
```

Then, move all your platform specific Make routines and definitions to a makefile with a name such as **Makefile.$ARCH**, where **$ARCH** is what is returned by uname on the particular platform. Finally, in your main makefile, add the following line to include the platform specific definitions in your make commands:

```
include Makefile.$(ARCH)
```

This results in the correct makefile definitions being automatically loaded by make at runtime.

The next stage in your build environment is finding a Java compiler that has been ported to Linux. At the moment, there are a number of Java compiler and interpreter projects in the pipeline. But so far, for production work, the options are the Java development kit that has been ported by the Blackdown team, or the JDK that is available from IBM. Depending on your situation, there may be preferences towards either JDK. Both these JDKs support the latest specifications from Sun.

Another advantage of using Java is the cross-platform nature of the byte code that your source code is compiled into. This means that if you need to use extra class libraries or jar files, there is no need to go hunting for platform specific implementations. However, many Java applications do in fact include native libraries, and this limits execution of these applications to the platforms that they have been ported to. Depending on how much you need to run these applications, this may not be a problem. For example, the project that I'm working on makes use of the Java Messaging Specification (JMS), defined by Sun. The Java Message Queue (JMQ) is a product released by Sun that implements JMS. JMQ hasn't been ported to Linux as of yet, and as a result, I am unable to test my code against it in Linux. I do need to compile my code against it, and because the libraries are mostly in jar format, I can copy the jar libraries onto my local machine and still successfully build my application against it.

Applications to help you develop code are two a penny. Each editor or IDE has their evangelists, so making a recommendation in this area is always a touchy subject, but I will mention my favorite editor. Due to job requirements, I was forced to learn **vi** a few years ago. Since then, I have moved onto **vim** and **gvim** and never looked back. I've tried other IDEs now and again, such as JBuilder and even Microsoft's Developer Studio, but until they implement the vi keymaps, those IDEs will only ever get a look from me. Where these IDEs do have advantages over vi is in the debugging aspect, but sometimes a well-placed println() method will do just as good a job.

Vim and gvim have color syntax highlighting, auto-indentation and other features that are too numerous to list. These editors can even handle write-build-debug, if set up correctly. An extremely powerful and easy-to-use function of vi clones is macros. A little thought about repetitive tasks in an editor can result in a simple but powerful macro to help you do what might have taken a lot longer in a less powerful editor.

What we're left with, then, are two of the more project-management type features that are usually left until last in any self-respecting project: source code management and bug/defect tracking. The most widespread source code management packages are SCCS, RCS and CVS. SCCS isn't available as an open-source application, which leaves CVS and RCS. RCS is suitable for smaller projects. I recently had the opportunity to work with CVS. While, for the most part, it uses the same format history files as RCS, the user interface to CVS is much more elegant. Added to that, you have multiple-developer support and remote-client support. That last feature comes in quite handy in my setup. Our CVS repository is stored on a Solaris machine. In my aim to use all things Linux and to prove that it can actually be done, I installed a CVS client on my Linux machine (this would probably come standard on some installations). By setting the CVSROOT environment variable to access the CVS repository on the Solaris machine remotely, I can manage my source code locally. CVS uses **rlogin** to execute the CVS commands remotely, so make sure that you have set up the proper access on the remote machine. The CVSROOT environment variable needs to follow the following format in order to have it access a remote repository:

```
export CVSROOT=:ext:hostname:CVSRepository
```

Good open-source bug/defect tracking software is hard to come by. I could only find two that looked at all stable. One of these was dropped because it didn't fully meet our requirements. The only real possibility available to us was the bug-tracking application used by the Mozilla team. This application has a decent web-based user interface, with a MySQL database back-end. It is highly configurable, and once we got over a few quirks, it was up and running reasonably quickly.

Utilities

Some of the typical Linux utilities come in very handy when developing a project. Not just for the work involved in coding and testing, but also the utilities that simply make life a bit easier. For example, the commands **find**, **cat**, **awk** and **egrep** tend to be used quite a lot for general system administration, finding your way around your source files, and writing small scripts that make a job easier. Without installing something like Cygwin for Windows, you would be hard-pressed to find similar “life-improving” utilities on Microsoft platforms.

A particular class of utility that is more directly related to code development would be a “pretty printer”, or utilities that can tidy your code for you so that they conform to a particular standard, such as the Sun Microsystems Java coding standard, or your own company's coding standard. The indent utility is available on most Linux systems, and by specifying a range of options, you can control how the final output is formatted. At the moment, our project conforms to the Sun Microsystems coding standard. When I first went looking for the options for indent that would format Java code to the correct standard, I came across the **Jindent** utility. This is an indenter written in Java, so again, it is completely cross-platform. By default, it formats to the Sun standard, and when creating your own configuration file, you can modify its output.

System Setup

For backup purposes, we keep most of our development work on UNIX servers. The majority of our client machines are Windows NT. The UNIX machines we work with are equipped with NFS and SAMBA. By sharing out these drives, we can access those resources locally without having to log into the remote machine. A Linux machine can mount both NFS and SAMBA drives that have been shared out. By keeping the same directory structure as that on the remote machine, you reduce the need to produce machine specific makefiles. For example, if a required jar library is located under /opt/FSUNjmq/lib/jms.jar on the remote machine, the makefiles that you developed can run on both the remote UNIX machine and your local Linux machine without any modification, by mapping /opt/FSUNjmq to /opt/FSUNjmq on your local machine. Commands for sharing out remote drives are similar to the following. For NFS:

```
share /opt/FSUNjmq
```

For SAMBA, edit the /etc/smb.conf file, copying one of the example shares, modify it to point to the correct directory, and make sure to give your user name access permission. It is not always possible to work solely off your local machine; there will be times when you will need to log in to the remote machine to run various processes, such as when there isn't a port available for the application you wish to test against, or when you need to copy, move or access large amounts of data. In addition, logging into the remote machine will give you better network performance. The **rlogin** command gives you easier access to remote machines than telnet. With rlogin, you can set up access permissions so that you are not required to enter a user name and password every time on login. By creating a .rhosts file in your remote home directory, and by adding your local machine name, you will be able to drop into the remote machine easily. Just make sure that your .rhosts is set to read/write permission for the owner only, and that there is no group or global access allowed. Otherwise, your automatic authentication will not work.

One of the annoying things I found about logging into remote machines is that you have to set up all your shell preferences a second time. This can be overcome by configuring a common profile file and making this available to both your local machine and your remote machines. Create a .commonProfile file on your remote home directory, and mount your remote home directory locally. So, in addition to your /home/username directory, you also have a /remote/username directory. In your local and remote profiles, you can then source your .commonProfile. Any changes you need to make can now be made only once, and they will be reflected in both environments.

One important point about remote access to your UNIX servers is that when setting up a user account for yourself on your local machine, you should set the same username, uid and gid that you have been assigned on the remote machine. This makes life much easier in terms of write, mount and login permissions. Some of the more experienced users may be familiar with NIS; this is a method for keeping a central repository of users and passwords across a number of UNIX machines. With a bit of research, and a suitable network configuration, it may be possible to set up your Linux machine to use NIS to allow other users to access your machine. This avoids the need to duplicate each user id and group id when you add a new user to your machine.

An extremely useful ability is being able to run X Window clients from the remote machine on your local machine. This gives you access to the remote machine in a graphical environment. However, even though you can run remote applications in a window on your local X Window display, in some cases there are incompatibilities between what the remote application is expecting from a UNIX X Window system and what your local X Window server provides. A classic case would be if your X Window server is running in 16-bit color, but the remote application can only run in 8-bit color. Rather than having to shutdown your X Window session and restart in 8-bit color, you can either start up a second X Window session with 8-bit color locally, or you can run the remote window manager so that it displays on your local machine. The command for this is:

```
X :1 -query remotehost
```

The “:1” means that you want X to run as your second display. This requires you to set the display variable on the remote machine to your second display:

```
export DISPLAY=yourhost:1
```

General Advice

In the way of general advice on how to handle day-to-day tasks that would normally be done in Windows, I offer the following:

Probably the most generic function in any office these days is reading and writing Microsoft documents, whether Word, Excel or Powerpoint. StarOffice provides an entire suite of applications that are capable of reading and writing Microsoft format documents. With their latest version, they have provided even more support for Microsoft integration.

For e-mail, there are loads of mail clients to choose from. However, this can be restricted depending on the type of mail server you use. You should have no problem finding mail clients for POP and IMAP mail. Since I've already mentioned StarOffice, I'd like to point out that it comes with a built-in mail client for POP and IMAP mail. Netscape also includes POP and IMAP support. Difficulties can occur if your company has selected to use Microsoft Exchange as its e-mail server and hasn't opened up IMAP or POP access to it. At the moment, there are no IMAP compatible clients available for Linux. As far as I know, your options here are limited to e-mail clients that will only run in Windows (if anyone knows any differently, give me a holler!). See below for more information on how to combat this.

Unfortunately, Linux doesn't supply all the capabilities that are necessary in today's office. We are more often than not resigned to having to share our machine with Microsoft products. **WINE** provides the ability to run Microsoft products natively in Linux; however, it leaves a lot to be desired. For these cases, VMWare is a clever application that allows you to run an instance of another operating system within your Linux operating system. Check it out if you haven't come across it. For me, VMWare provides the ability to run Microsoft Outlook reading e-mail from our Exchange server. It seems a bit of an overkill to have another operating system running in order to just read mail, but we don't always get what we want.

Printing is an area that can cause a number of headaches when setting up a Linux operating system; however, printer setup tools and driver support is improving rapidly. Setting up a network printer under Linux can be pretty straightforward if you follow the rules. One thing to watch out for is when you have a printer that doesn't seem to have its own driver. There are plenty of resources on the Web detailing how to set up a specific printer even if it doesn't seem to be supported. For example, we have a Gestetner PCL printer in the office, but by using the HP LaserJet III printer driver I now have access to this printer.

In summary, there are many areas that must be dealt with when trying to successfully run a Linux operating system in an environment that doesn't necessarily support it. I've tried to give you some idea of how I got around the problems I've encountered. In the Resources section, I've listed some of the web sites that have come to my aid in researching how to do something in Linux that is normally done in Windows. But remember, not everything can be done through Linux, and sometimes you will have to fall back to non-Linux applications.

Resources

Mark currently works for ICL, based in Dublin. His technology interests include Linux and Java. Outside of work, he likes to take long holidays to visit other countries. He can be contacted at mark.stacey@e-merge.ie

Archive Index Issue Table of Contents

Advanced search

Linux and Networking: The Next Revolution

Marcio Saito

Issue #79, November 2000

Recent changes in the areas of both software and hardware are combining to revolutionize networking.

Revolution is defined as a sudden and fundamental change, and having seen the effects of one revolution does not always allow us to foresee those of the next. Revolutions can cause new waves of innovation and shifts in power and control, changes that are unnerving and exciting at the same time.

One might say that the computer world is experiencing revolutions in the areas of both hardware and software. What might happen in the networking arena when they intersect and amplify one another? Of particular interest is how possible changes will affect the functionality of the solutions available to endusers, now that it is suddenly possible to build a new generation of network appliances that are more powerful and incorporate richer functionality than traditional router appliances and other networking equipment. The potential impact of Linux on networking is even more significant than the impact it has already had on the server market.

During the 1970s and 1980s, computer networks were predominately host-based, with mainframes or midrange systems constituting most of the processing power. Users were directly connected to the host using simple, non-intelligent terminals.

In the early 1990s, computer network protocols converged into a few standards and were adopted for use with desktop computers. Local area networks became a requirement for almost any type of business. While network services were distributed among many servers, the network connectivity migrated from host computers to specialized hardware designed and built to perform internetworking functions. These specialized network appliances (routers, switches and access servers) allowed more reliable, cost-effective and efficient networking. Network connectivity (provided by a router) and network services (running on the servers) are seen as separate entities. This is how most people understand networking today, but cheap hardware and open-source software is beginning to change this.

The PC revolution that consolidated in the late 1980s extended computing power to almost every office desktop. Fueled by a high level of competition and the establishment of industry-standard hardware and software, PCs became more affordable and more powerful. With the Internet, an explosion in the demand for home computers was triggered. PC manufacturers were able to leverage the volumes driven, and prices of PC-related hardware dropped sharply. An article in the May 1995 issue of *PC Magazine* said, “As of spring, the touchstone price is $1,999 (US) for a Pentium/75 multimedia system with 8MB RAM, a 700MB hard disk, and a 15-inch monitor.” In January of 1997, when the first PCs for under $1,000 were offered, the same magazine wrote, “So what does $999 get you? You can buy a 120MHz or 133MHz system, for less than $1,000.” As we start a new decade, consumer PC prices have dropped further. Today, we can buy desktop computers with CPUs running over 500MHz, 128MB of RAM and many built-in peripherals for a few hundred dollars. That is about the same price you would pay for a typical access router with much less impressive hardware specifications.

Because the architecture of servers and desktops are similar, manufacturers can take advantage of the low cost of components to build inexpensive server systems. Although cost was one of the important factors in the substitution of servers for routers, this is no longer necessarily the case. Standard hardware components are becoming so inexpensive that it is almost impossible for the manufacturer of a proprietary hardware device to be competitive. This trend is reaching a threshold where the addition of a catalyst could trigger a paradigm shift.

Linux may be that catalyst. Distributed without restrictions on use and installation, it is always provided with source code. It is not necessarily free (zero cost), but anyone can change or improve it to meet specific requirements. Frequently, those requirements are shared by others, and the changes or improvements are fed back to the community.

According to the latest IDC numbers (August 2000), Linux was the second most popular server OS in 1999, with 24% of new server licenses, and is the fastest growing one (the Windows platforms together have 36%). The favorite for Internet-related applications, it is growing quickly in the enterprise market for corporate applications. Due to its roots in the Internet, Linux developed strong networking support, better than other commercial operating systems. Because it is open source and receives the contributions of a huge developer community, Linux is more flexible and evolves much faster as well. The Linux operating system has features, security and robustness comparable to a specialized, internetworking operating system. Put it together with commodity hardware and the result is a very powerful network platform.

It's obvious that Linux is changing the server market landscape: 24% of the market is substantial, regardless of your preference. While it remains to be seen whether Linux will change the client/desktop market, its impact on the networking market is sure.

In this early stage of the revolution, there is still a need for technology integrators to make these benefits widely available. Some technical users are doing the integration themselves. They get communication boards, integrate them with standard PC hardware, and build their own Linux-based network boxes. One such example is Internet Service Providers who, instead of buying a PPP remote access server to provide dial-up Internet access, use Linux servers with multiport serial boards connected to modem banks to perform the same function. Some technology integrators, however, are already delivering a successful new generation of network appliance products. For example, the Cobalt Qube is an all-in-one Internet gateway for small-and medium-size businesses, that can be fitted with a routing board for WAN connectivity. The whole solution integrates all the Internet functionality needed, including network services and connectivity, is very easy to set up and manage, and costs about the same as the less functional access router it replaces.

But this new network device is not simply a more affordable replacement for the traditional router. It has the added advantages of expandability and flexibility. As routers were once better adapted for the network of the past, the new network appliance is better adapted to the realities of the future.

Users end up getting new products that are cheaper, better and easier to use; products that replace the router or access server, incorporate new networking services, and can be easily customized to each different application. It is a different kind of product.

To fully understand the impact of Linux in this mix, it's necessary to consider the latest IDC numbers (available at www.idc.com/itforecaster/itf20000808.stm). The total client and server OS market was about $17 billion in 1999. Windows generated almost $8 billion in revenues, while Linux generated less than $100 million. Considering that Linux now has a substantial share of the market, those numbers are shocking. From the user standpoint, the value of a solution is the same, independent of the OS being used; so one would expect revenues to be proportional to the market share. If Microsoft is making $8 billion on Windows, where are the Linux revenues? Because Linux combines open architecture and the business models implied in the open-source model, its “revenues” translate almost directly into savings for endusers—savings that can be used to pay for integration and services that produce better solutions for each user.

This gives some idea of the impact and the shift in control and power that Linux brings to the table. But our focus is networking, rather than general purpose OS. According to the last Data Communications' market forecast, the size of the network equipment market in 1999 was $70 billion in the U.S. and $120 billion worldwide. All of this money is going to the equipment manufacturers, holders of proprietary software and hardware technology, such as Cisco and Nortel.

So, when open architectures replace proprietary boxes, a lot of money will change hands. In the networking market, this is happening in both software and hardware at the same time, the impact is amplified.

In the past, proprietary solutions were used because they were cost-effective compared to server-based solutions. Linux and standard hardware frees the market, allowing technology integrators to produce better solutions and be competitive without the need to drive large volumes (the large volumes are already integral to open-source and standard hardware). The need for better solutions drives the change, and open-source and commodity hardware enables it.

These changes have important consequences. Today, users depend on proprietary networking box solutions for features and functionality. They work well for connectivity but cannot have services added or be customized in terms of functionality. Users are driven by economics to separate network connectivity from network services, and to choose solutions that are many times more cumbersome and difficult to manage. In the near future, however, additional functionality will be incorporated into the connectivity product, and using an open platform will make incorporating new hardware or software technologies simpler and quicker. Users and technology integrators will not depend on a sole technology provider, and control will shift toward the enduser.

As with any change, at first it is not easy to perceive all the benefits of a new approach. For networking, all the elements are in place and changes are coming. New possibilities will be discovered along the way, and those discoveries will lead to a new, more powerful approach to networking.

Marcio Saito (marcio@cyclades.com) is director of technology for Cyclades Corporation.

Archive Index Issue Table of Contents

Advanced search

Dissecting the CueCat

Michael Guslick

Issue #79, November 2000

Cutting this scanning kitty up to to discover the hideout of the serial identifier.

I picked up a CueCat bar code scanner at RatShack (“You've got questions. We've got blank stares.”) yesterday along with a few other odds and ends. [see http://www.getcat.com/ to get your own free CueCat—Editor.] If the store actually carried the stuff in their commercial sales catalog, I'd be a happy camper—I don't need an animated Elvis Presley phone, I need a local source of microcontrollers and specialized ICs. But I digress.

One of the things that has a few people worried is that the clerk at Radio Shack takes down your name and address in their system before giving you a CueCat. However, there doesn't appear to be a way of tying a particular CueCat to a person at the time of purchase (although Digital Convergence can most likely trace a CueCat back to a particular Radio Shack). Although each CueCat has a unique serial identifier, each CueCat package has the exact same bar code on the front (which is what the clerk scans in). My goal was to find where that serial identifier lurks inside the CueCat....

I opened up the package and immediately began to open up the scanner (see Figures 1 and 2).

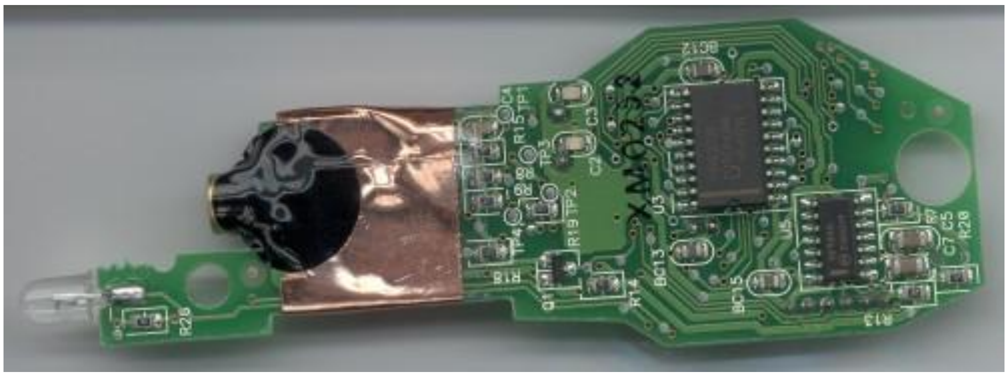


Figure 1. Scan of the Bottom Side of PCB

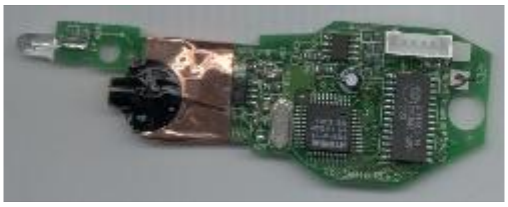


Figure 2. Scan of the Top Side of PCB

I carefully removed the shield and forcefully removed the serial EEPROM (see Figure 3).

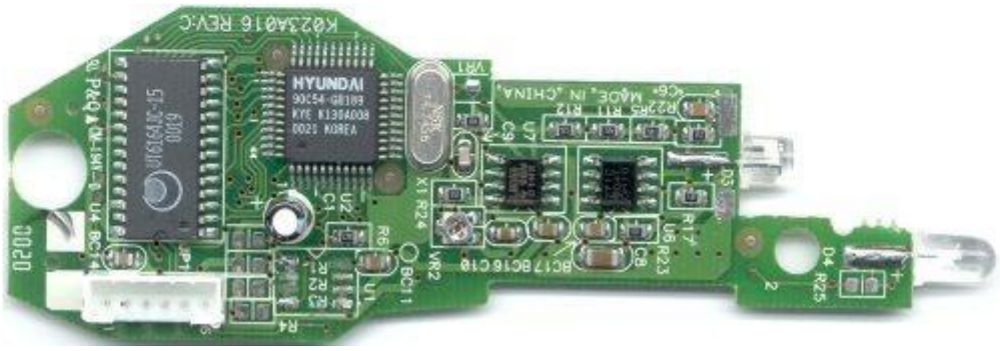


Figure 3. Image of the Board with the Shield Removed

The component that caught my eye was the small 8-pin device (U1) on the top side of the board (see Figure 4). For a detailed list of the semiconductor devices see the sidebar.

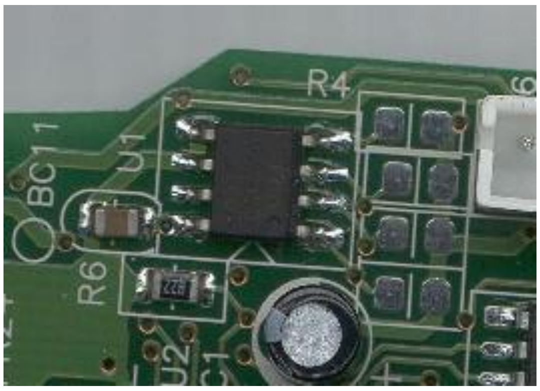


Figure 4. A Small 8-Pin Device

Semiconductor Devices of the CueCat

The scan doesn't really show the markings at all, but it's an ATC 93LC46, which is a 1kbit serial EEPROM. Unfortunately, ATC doesn't have datasheets for the device available on their page. Not to worry, as other manufacturers, such as Microchip and Holtek, have 93LC46s available. The datasheet for Holtek's HT93LC46 is located here, and it's a closer match than the Microchip unit, as it implements an ORG pin to control how the memory is accessed (in the above picture, the ORG pin is tied to VSS—this would make the unit addressable by

128 8-bit words if it was actually a Holtek 93LC46, but the ATC unit appears to be set up the opposite way—more on this later).

The first thing I tried was removing the 93LC46 from the board. However, I'm really not equipped to desolder SMT devices, so this was rather futile. So, I simply soldered some wirewrap wire onto the pins to see what's going on. I hooked my trusty scope up to them and found that the data is read out of the 93LC46 only on power up of the CueCat (about 100ms after power up, to be exact).

After this, I tried hooking up the 93LC46 to a PIC microcontroller (with a little bit of code that I whipped up) to see what lurks inside the serial EEPROM. Unfortunately, I managed to wipe the contents of the EEPROM (looks like it reads back all locations as 0xFF now). Oh well, at least I don't have that pesky serial number in there anymore.

Unfortunately, hooking up a microcontroller to erase the EEPROM is a little out of range for your average privacy-concerned individual. I'm guessing it should be possible to disable the serial number by cutting the CLK line to the EEPROM, which should be easy for anyone with a keen eye and a sharp X-acto knife.

I'll have to wait until I get another CueCat before investigating what's inside the EEPROM. In the meantime, I've been looking to find exactly what EEPROM data areas are being scanned. With my trusty Tek TDS 210, I took a closer look at the CS (chip select), SK (clock) and DI (data input) lines. For those interested, the 93LC46 is an SPI (Serial Peripheral Interface) device—it uses a synchronous serial line to transfer data (your computer's serial port is asynchronous and doesn't use a separate clock line). The CS line is used to tell that particular chip that it's being talked to, otherwise it will ignore data being sent to it.

The 93LC46 is sent a total of nine commands (they are all read commands, but more on this later). The CS line goes high a total of nine times (the first CS <\#145>pulse' is extremely long, as the CS line goes high as soon as the CueCat is powered on), and I used this as a baseline to see what was happening on the SK clock line (since my scope has only two channels, I can't look at every pin at once). I noted that there were 27 clock pulses during each CS “pulse” and I could see a gap between clock pulses where the CS line went high-low-high. I hooked up the SK and DI lines to the scope and took a look at exactly what bits were being sent: CS “Pulse” Data Clocked In

1 0110000001111111...2 0110000010111111...3 0110000011111111...4 0110000100111111...5 0110000101111111...6 0110000110111111...7 0110000111111111...8 0110001000111111...9 0110001001111111...

Okay, now the first thing to note is that the leading **0** is basically garbage, as the first **1** is really a start bit (and not yet the beginning of a command). Also, the trailing 1s aren't really bits sent to the EEPROM—these are clock pulses provided for the EEPROM to write out its data on the DO line. So what we really have is a command like this:

```
10000110
```

followed by a high DI line. The first two bits are the command, followed by the address. In the 93LC46, **10** is the read command. But what's this? We only have six bits to define the address and a lot more than eight clock pulses after the command is sent—the EEPROM must be organized as 64 16-bit words!

So, the microcontroller reads in a total of 9 16-bit words from addresses 0x01 through 0x09 (I have no idea why they didn't start at 0x00). Note in this sample scan,

```
.C3nZC3nZC3nYCNr2C3fWCNnY.fHmc.C3DZCxPWCNzWDNnX.<\n>
     000000001175023101      UPA     040293153502
```

that the serial ID field is 18 characters long (or 9 16-bit words). I wonder if they're hiding anything nifty in the other 55 words? And why did they use a serial EEPROM? I would think that something like Dallas Semiconductor's silicon serial number would be a smaller, cheaper, totally non-volatile alternative, but maybe this gives DC better control over assigning IDs (perhaps there's a “special” bar code that can be scanned in to rewrite the EEPROM?). I'm glad they used a 93LC46, though—you can desolder them and use them for other stuff...

Anyhow, I'm itching to try disabling the serial ID by cutting one of the traces to the 93LC46—I don't have a virgin CueCat to try it on, but if anyone wants to give it a shot, cutting any of the traces shown by the yellow cut marks, indicated in Figure 5, should disable the serial number (or give floating voltages to really whack out what the microcontroller is reading back—you may even be able to get some <\#145>random' serial numbers generated this way).

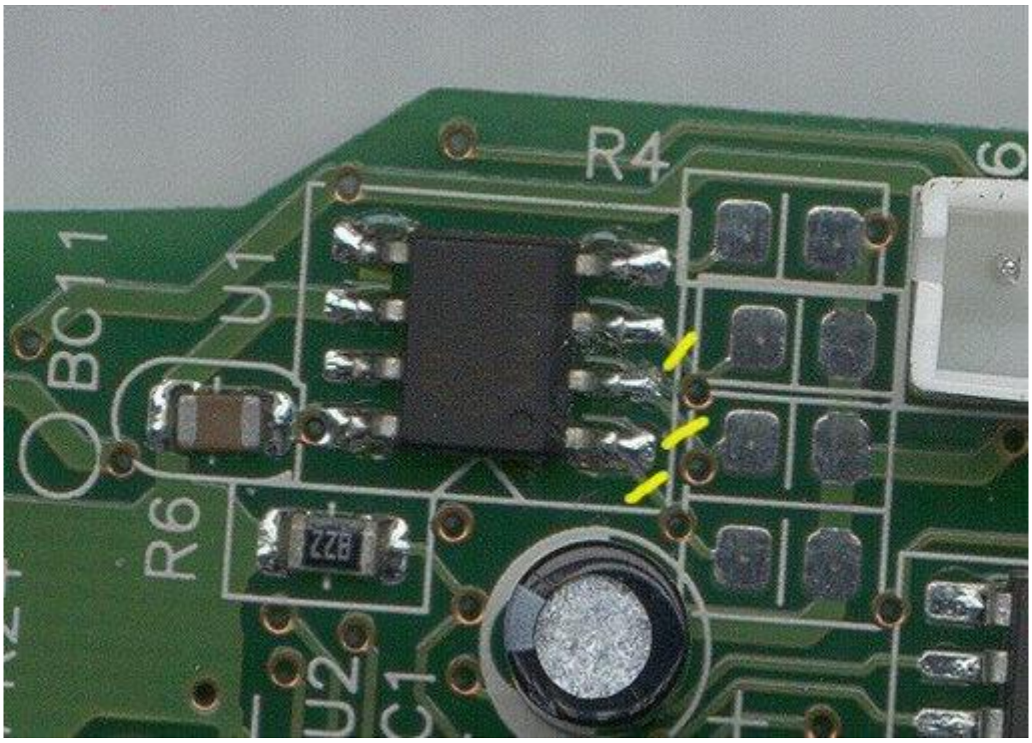


Figure 5. Cuts From Top to Bottom Disable the DI (data in), SK (clock) and CS

Or, you can slice the line, indicated in Figure 6, by the microcontroller to sever the DO (data out) line (I'd be inclined to try this one myself—the floating voltages could be fun here). Remember—you should need to cut only one line to disable the serial ID—take your pick.

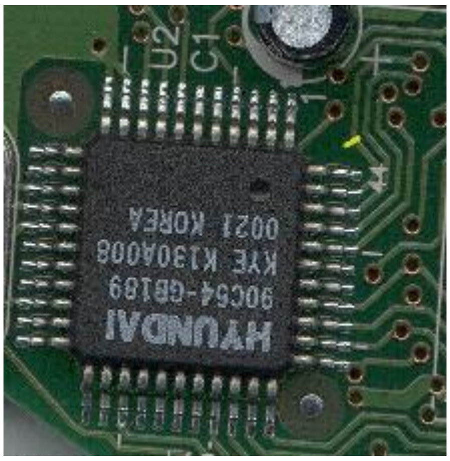


Figure 6. Cut Here to Disable the DO (data out) line.

If anyone decides to give it a try, let me know how it turns out—your CueCat should still be able to read bar codes without any problem.

I picked up another CueCat last night—this one is the 68-1965-A model (supposedly more common) rather than the 68-1965 which is shown here. I'll be tearing apart this one later and posting the innards. How can you tell the difference? The A model has four small screws holding it together, the older one has two larger screws. The A has a small grommet for the wire on the cat's butt and a large black square for the scanning window, rather than the smaller rectangular opening on the 68-1965.

Oh, and if any lawyers representing Digital Convergence want to send me threatening letters, cease & desist orders, or more hardware to disassemble (I'd appreciate a USB CueCat when they become available). Note in advance, however, that any legal mumbo jumbo will be met with a polite “s—w you!”

Michael (mguslick@matrixpm.com) graduated with a degree in mechanical engineering from the University of Wisconsin-Milwaukee (where he was first introduced to UNIX and of course, Linux). He enjoys computers, electronics, and spending time with his fiancée, Kristin. He is also hopelessly addicted to playing paintball and squanders vast sums of money on the sport. On-line, he goes by the moniker of “Have Blue”.

An Interview with Red Hat's Michael Tiemann

Dan Wilder

Issue #79, November 2000

Last year, Red Hat bought Cygnus. In this interview, take a look at how embedded systems have become an important part of today's Red Hat.

Red Hat Chief Technical Officer Michael Tiemann is well known for his extensive and important work on **g++** and **gdb**. A founding member and former CTO of Cygnus Support, Michael joined Red Hat when they purchased Cygnus. He now serves as Red Hat's CTO.

LJ: Tell us about Red Hat and Embedded Systems.

Michael: Let me start first by talking about one of the keynotes I've been giving this year. It is based on a book called *Guns, Germs, and Steel* by Jared Diamond, and I don't know if you've heard of this book. It won the Pulitzer Prize in 1998. He is a sort of an anthropologist and linguist and biologist all wrapped into one. He uses his extensive array of scientific techniques to answer interesting questions like, “Why did people from Spain set sail to the New World and conquer large civilizations instead of the other way around?”

With respect to embedded systems the question that I'm thinking the most about today is, “Will open source change embedded, or the other way around?” The reason that this is an interesting question is because the dynamics that open source enjoys-readily downloadable software, a large developer community sharing software which can easily be downloaded, installed and modified-is quite different than what we think of in the traditional embedded systems world where things like ROM are where programs are stored. The interesting question will be, “what extent are the constraints of the embedded systems world going to change the open-source approach to programming?” There's a lot of interesting stuff going on in that direction. Or, the other way around: is open source actually going to change the way people design embedded systems. I could give you a couple of examples of both.

One is eCos, the Embedded Configurable Operating System. eCos was designed to fit the constraints, whatever they were, of embedded systems quite efficiently. What we provide with eCos is a configuration technology which enables source-level configuration. What this means is that instead of developers needing to touch actual source code to make it conform to their world, they can use a tool to control over 200 different configuration points. The consequence is that you can deliver a real-time operating system whose memory footprint ranges in size depending on the needs and the assumptions of the application.

There are some applications which have already implemented their own cooperative threading model and have already implemented other resource allocation routines similar to those provided by today's conventional real-time operating systems. In the case where the application does all the work and they simply need a hardware abstraction layer, eCos can deliver that with less than 1,000 bytes of memory footprint. But if you want to turn on scheduling as a function of the kernel, or you want to turn on management of memory, or you want to turn on the existence of a file system, et cetera, et cetera, et cetera, you can do all of this incrementally and largely independently, and scale that up to typically 100 or 200 kilobytes at the high end.

We have customers who have not been able to configure conventional proprietary real-time OSes smaller than about 55 kilobytes. In eCos in this exact customer situation, where the customer's design requirements were 24K of memory footprint to fit in the on-chip ROM, eCos came in at 11K by utilizing its source-level configurability.

So, in the case where the requirements of embedded systems are changing the shape of open source, we have built shape-changing technology for eCos.

LJ: Is that shape-changing technology actually part of the base open-source release of eCos?

Michael: Yes it is. We announced the open-source availability of CDL, the Configuration Description Language, I think last month.

Now, putting the shoe on the other foot, and looking at how open source is changing embedded, I have seen a tremendous shift in interest from traditional proprietary real-time operating systems, to using embedded Linux. In fact, I can tell you that up until September of last year, the customers that I talked to tended to profile the exact same way that IDC profiled the entire market. In other words, X percent were interested in proprietary RT/OS “A”, and Y percent were interested in proprietary RT/OS “B”e and my own statistical sampling of the market when I talked to customers corresponded very accurately to what IDC was reporting in their vendor numbers.

Since the announcement of commercially available embedded Linux was made by a number of companies in September of last year, I have spoken with exactly zero people who are designing new embedded devices using proprietary operating systems.

LJ: That's quite a sea change.

Michael: That is. That's not to say that people are not buying these operating systems for legacy applications. I continue to meet with a lot of people who complain about how difficult it is to change. But in terms of new designs, it has been like night and day.

LJ: Does that include WinCE and PalmOS?

Michael: I have not talked to anybody, ever, who has been interested in WinCE. I don't know about Palm.

I would say that if you're simply looking to replace a proprietary RT/OS with embedded Linux, just to change the RT/OS in the socket, that's not very interesting. What's really interesting here is if you recognize not only the development community that comes with Linux, but also the extensibility and the openness of the platform. With the ability to create new kinds of computing devices that very much and very effectively blur the lines between the traditional desktop model, which is sort of going away, and a pervasive post-PC model, which is coming to the fore, it looks like embedded Linux may very well be the ingredient that has been missing in this new post-PC world.

LJ: Some of the developers I know are feeling that they're on a treadmill, keeping up with the accelerating change in hardware, and one of their problems is constantly having to rewrite device drivers to adapt to the new chip that replaces the one that's no longer available. I understand that one of the attractions of Linux is that other people are writing a lot of the device drivers.

Michael: Yes. Other people are writing the device drivers. But also, it is a common practice for people who write device drivers for Linux to make those drivers open source.

LJ: Exactly. Is the same dynamic happening with eCos?

Michael: Certainly. That is a dynamic which we're increasingly seeing. eCos went through what many real-time operating systems go through, which is that we announced it two years ago at Embedded Systems West in San Jose. It went through a gestation period. Of course there are always the innovators and early adopters who pick up stuff aggressively, but by and large people waited to see whether or not eCos would mark its first, and now its second, anniversary. One thing that we've noticed is that as we are marking its second anniversary, the number of designs has increased exponentially. With those designs comes an increasing number of participants in the development community. That means new board-support packages, ports to new architectures, device drivers, et cetera. In other words, it is really beginning to reach its critical mass.

LJ: So what kind of quantities of “design wins” are you looking at these days?

Michael: It's a little tricky. You could be on the list of people who will get the pre-briefings about announcements we're going to make at ESC West. But I can't spill those beans too much.

LJ: I'm just looking for a ballpark. Ten solid wins, a hundred?

Michael: I think we've got in the neighborhood of tens of announceable wins.

LJ: Could you compare and contrast the areas of application of embedded Linux with that of eCos?

Michael: eCos is really for devices that want to have a very simple footprint, the sort of classic embedded system design where you have a microcontroller and a device to be controlled. The benefit that eCos gives you is the ability to use open-source availability, the ability to squeeze a particular design into much smaller spaces than would be practical with Linux.

One that we did announce previously, for example, is an operating system for Brother's 2400cen printer family. The jobs of these printers are to be peripherals that apply ink to paper. It is not likely that users are going to want to be running high volume web servers on their printers. It is also not likely that people are going to want to use their printers as additional computing devices for other purposes. Of course, I'm sure the SETI@Home guys would love to double their listening capability by running on unused cycles of color and black-and-white printers. But seriously, eCos is much more designed for purpose-specific devices.

Linux is appearing to be a real runaway success story in the space of Internet appliances, both in the server and client space. Last month we announced a design success with Ericsson, who has actually demonstrated a screen-phone platform running on top of Red Hat Embedded Linux. Our hope is that we will have at least a picture, if not a model, of this phone on display at the conference. This is Ericsson's bid for a new, richer, completely connected device on the client side. There's also a lot of activity on the server side.

There are server appliances, residential gateway servers, small office/home office servers, where you don't want to be running a full desktop distribution of Linux, and you don't necessarily want to be running it as a complete Linux server, but there are capabilities you want, like web serving or firewall or e-mail or file serving et cetera. The reason you want to have these devices configured more narrowly is because it dramatically reduces the complexity of administration.

LJ: Bring it home and plug it in.

Michael: If you want to do something different, you can just download a different profile and it will have those properties. When you have a normal Linux server doing lots of different jobs, it increases your need to have people who understand network security, and it also means you have to make more complex resource allocation and monitoring decisions.

LJ: With freedom comes responsibility.

Michael: Exactly.

LJ: You mentioned Red Hat Embedded Linux. Is that a distinct distribution at this point?

Michael: It's a toolkit. It uses our graphical user-interface technology, called Source Navigator, as a front end, and what we have right now is a standard Red Hat distribution configured by this graphical user interface. So it's not source-level configurability, like we have with eCos, and in fact it is an open question whether the embedded systems market will change Linux to be more supportive of source-level configuration. The current thinking is “no”, but thinking about the future is uncertain. If we don't have source-level configurability, then we still have module-level configurability, and that's what the graphical user interface from Red Hat offers.

LJ: Is that a distinct Red Hat product?

Michael: Yes. We call it the Embedded DevKit. You can actually get it off our web site for $199 (US).

LJ: What's happening with your embedded API proposal, EL/IX?

Michael: Everywhere I talk about EL/IX, I talk about POSIX 1003.13. I gave a couple of talks about embedded Linux at LinuxWorld, and I talked about the fact that we announced EL/IX two weeks before the POSIX committee went public with their 1003.13 recommendations. We had known that they were brewing this, but we also had no certainty as to what the timeline would be. Because we knew what they were doing, and because we liked their approach, we took what we figured to be their ideas. We said, okay, there are going to be four profiles, and they are going to have these bundles of functionality.

Then what happened is 1003.13 came out, and PE51, PE52, PE53, and PE54 corresponded to the four levels that we had picked. Had POSIX come out with thirteen profiles, or twelve, or seven, or two, then our work wouldn't have looked at all like theirs. But the fact that they chose four profiles, as did we, and that those four profiles do have a pretty straightforward alignment, means we're getting behind POSIX 1003.12 as a recognized international standard, and we're lobbying to incorporate more of what we consider to be the true embedded developer's viewpoint. For example, POSIX is very proud of their signal model. But, not a lot of embedded developers that I have talked to like that signal model very much. POSIX signals are not an option in these profiles. However, they are an option, in or out, in ours. The way to look at EL/IX is four profiles: from minimal controller to minimal real time to single process to full-blown multiprocess system.

Number one, we think the theory is correct, but we want to offer people additional configuration options. Whether or not you have signals, whether or not you have networking support, whether or not you have file system support, is (or can be) orthogonal to the four levels.

LJ: So your energy at this point is going behind the POSIX effort, and bringing that forward and making it useful to embedded developers?

Michael: Yes.

LJ: Then the process of EL/IX implementation, either as a layer on top of eCos or as additional elements in the Linux kernel, needs to wait until some finalization on the POSIX 1003.13?

Michael: No. The thing that needs to wait is whether or not Linux is going to support the concept of source-level configurability. As far as what's in POSIX

1003.13, and as far as what options and configuration bundles we want to offer, we believe that our approach is consistent with what POSIX is doing. We anticipate that we will offer something which will be POSIX-conforming. It will also have other options. The fact that you can always get to POSIX from Linux is going to be something which will make people very comfortable.

LJ: So you're moving your efforts forward anticipating converging with POSIX as it moves along?

Michael: Yes. I think that last year Linus Torvalds said he's happy to see POSIX become more Linux-compliant.

LJ: Your EL/IX process is moving forward at this point?

Michael: Yes.

LJ: The EL/IX draft standard 1.1 on your web site is dated in January. Is there going to be a new draft?

Michael: I am hoping so. We've got people under the gun for the Embedded Systems Conference. I can't make any promises. But we'd certainly like to have something fresh for the September show.

LJ: How's the Red Hat/Cygnus integration going along?

Michael: It's going along very well. The magnitude of the merger was pretty substantial. We hit the ground running at the beginning of this year, and in another ten days we'll be announcing earnings. Last quarter when we did our earnings announcement, there were some very clear ways in which Cygnus enhanced the Red Hat business with key customer wins, and since then we've publically announced the relationship with Motorola for high-availability embedded Linux servers, also with Ericsson on the screen phone. We've probably got some other announcements coming out.

If we were a private company I could tell you a lot more exciting details.

LJ: You mentioned Hitchiker's Guide to the Galaxy in another recent interview. Who's your favorite character in that book?

Michael: I think it would have to be Ford Prefect.

LJ: Why would that be?

Michael: Because he is so oblivious of his own absurdity.

LJ: I guess that would be a good model for most of us.

Michael: I also enjoy his optimism.

LJ: Thanks so much, Michael, for taking the time to talk with us.

Michael: Thank you, Dan, and I hope I see you down at ESC West.

Resources

Dan Wilder is technical manager at SSC. He can be reached at info@linuxjournal.com.

It's Mod. It's Layout. Any Questions?

Brian Aker

Issue #79, November 2000

Web composition solutions provided by ModLayout.

Like most adventures into open source, **mod_layout** began as a need. Around 1997, I was working on a web site called the Virtual Hospital (http://www.vh.org/). At the time, we were using Netscape's Fast Track server. I really wanted to switch over to Apache but had to overcome two hurdles. First, we had to get management to accept that it would be easy to find people who knew how to configure Apache (Netscape's graphical interface seemed to create a belief that it would be easy to configure and maintain) and, second, Netscape could do footers. The Netscape server supports footers by allowing you to arbitrarily add HTML to the end of any document that it is processing. This is far from perfect, but it serves the basic needs of most sites.

The first hurdle was easy to clear, seeing how we had to edit the Netscape server's configuration file consistently, and with so little documentation available on its file format, we found that maintenance quickly became a chore.

The second problem was solved by creating a custom handler. The solution was a small application written in C that was called for every HTML document. While this worked, it had a number of problems. For example, we now had an application being run a couple of times every second. This also didn't solve the problem of our CGI scripts, which still had to be modified so their output matched that of the HTML handler. This was far from an ideal solution.

I kept running into similar problems as I worked on different web sites. Someone would come along and want copyrights, headers or something similar applied to all documents on their site, and, typically, they would present me with yet another new web site technology for creating content. They would also want to do this without changing the content of their site, and it had to be easy to update and modify.

To answer these demands, I created **ModLayout**. ModLayout determines whether to handle a document by looking at the handler and mime type. For instance, if you wish to handle PHP4 scripts, you would use the directive **LayoutHandler** with the argument **application/x-http-php**. ModLayout handles most of the common MIME types without the need to specify the handler in Apache's configuration file; these can be disabled with “LayoutDefaultHandlers Off”, and you can then add back in what you want to have wrapped. Often sites will have specific documents that they don't want wrapped. For this, you can use the directive “LayoutIgnoreURI”, and there are also ignore directives for removing only the header, footer or HTTP header.

ModLayout provides three additional stages to the Apache process. It provides a header stage, a footer stage and a stage for doing HTTP headers. If you need to create HTTP headers dynamically for all documents, ModLayout will let you do this. Headers and footers occur on the HTML page before and after the original document.

There are three types of headers and footers. You can either add a simple piece of text:

```
LayoutHeader "Sample header"
LayoutFooter "Sample Footer"
```

or you can insert entire files by using:

```
LayoutHeader /usr/local/apache/htdocs/header.html
LayoutFooter /usr/local/apache/htdocs/footer.html
```

or by URI:

```
LayoutHeader /header.pl
LayoutFooter /footer.php
```

The above examples will work when you have simple requirements for headers and footers. The first two examples also work well because their content is cached in the Apache process. If you need something that is dynamic, however, you can use the URI example. ModLayout does not care about the type of URI used, it can be anything from simple CGIs and SSI to complex applications written in Perl, PHP or Java. Custom modules written in C using the Apache API can also be used. If you desire to just use only HTML but not have it cached, you can simply specify its URI and not its full path. ModLayout also provides additional environment variables for URIs being called as headers and footers. These supply information about the state of the original document. One of ModLayout's great advantages is that it works with any form of content and is language neutral.

A common use of ModLayout is to attach a copyright notice to all of the documents on a site. To do this, you could use the following:

```
<VirtualHost www.example.com>
     <...insert normal content..>
     LayoutFooter "<P>Copyright Example.com 1995-200"
     LayoutHandler "application/x-httpd-php"
</VirtualHost>
```

The above would insert the text at the bottom of every document on the www.example.com virtual host for all of the default handlers, including HTML pages and CGI scripts, and also on any PHP4 script present.

If you want to tack on the date the original document was last modified, you could insert the following into a PHP document, and save it as last-modified.php.

```
<?php
     echo("Copyright 2000 FooBar INC. <BR> Last Modified");
     echo ($LAYOUT_LAST_MODIFIED);
?>
```

and change the LayoutFooter directive to /last_modified.php.

Now, let's look at a more complex problem. Let's say you wanted to create a custom look and feel for a given site that was enforced by the web server, the following would serve as an example:

```
<VirtualHost www.example.com>
     <...insert normal content..>
     LayoutHeader "/header.php"
     LayoutFooter "/footer.pl"
     LayoutIgnoreHeaderURI "/index.html"
     LayoutHandler "application/x-httpd-php"
</VirtualHost>
```

In the above example, the site would have a custom header created by a PHP script and a Perl script running as the footer. All default handlers are wrapped along with any PHP scripts that are on the site. Notice that the index.html document will not be wrapped though, since most sites have a custom front page but want to have a default look for internal pages on the site. You can also use something like mod_random and its random quote feature to create your own banner ad system.

When using mod_random, the thing to remember when creating headers is that you need to be careful not to create invalid HTML (which, thankfully, most browsers will handle). But, if you are enforcing a look and feel on pages that outside users are creating, they can easily end up with a messy output.

Keep in mind that inserting any HTML before a page with frames will break the frames. At the moment, the solution most people employ is to use footers and/or create floating Javascript boxes). In a future version, you will be able to turn off headers, footers and wrapped content by directives received from a URI that is called from a header or footer.

For sites that have a more complex need, see Listing 1.

Listing 1

The example provided in Listing 1 is currently used for http://www.tangent.org/. This allows for web site document wrapping which affects the main document root content but leaves user's home pages alone. While the entire site uses the same footer, the header has been changed for different sections. **ModRandom** is used for inserting random quotes into the footers (thankfully these are just from *The Simpsons* and nothing annoying, like an ad banner).

Controlling ModLayout with Directory, Virtualhost and Location tags, allows you to work out some fairly complicated logic. You can also control what creates the HTTP headers through **LayoutHTTPOverride** and by creating your own dynamic HTTP headers with **LayoutHTTPHeader**. At the moment, there are about 24 total directives and ModLayout is under fairly active development, (developer releases occur about once a month and full releases about every two months). Feature suggestions are encouraged, and there is a mailing list for discussing features, HOWTOs and bugs. For more information, check out the Modlayout home page, http://tangent.org/mod_layout/.

Although ModLayout began as a solution to a problem that I kept running into, today it is very much the result of people writing in and requesting new features. It's future direction will pretty much depend on what people ask for and need.

Brian Aker is "getting paid to work on a BBS. Imagine that." The BBS is the famous http://www.slashdot.org/, where he is a "database thug and Apache guy." He is also the author of **mod_random** and other Apache modules.

Archive Index Issue Table of Contents

Advanced search

Real Hard Time

Doc Searls

Issue #79, November 2000

Real Hard Time

MontaVista's announcement was not met with universal approval. Both its most direct competitors, Utah-based Lineo and TimeSys Corporation had the following responses:

> Through our acquisition and integration of Zentropix, Lineo has been delivering hard real-time Linux for more than a year now. We have real customers using our hard real-time solutions today in the areas of flight simulation, weather-monitoring systems, heart-monitoring systems, industrial controls and many others.
>
> The Lineo real-time solution achieves guaranteed hard real-time microsecond response times today. This is very different from the “relatively fully preemptable kernel”, announced by this competitor.
>
> Lineo continues to actively support and promote real-time technologies. For more information about Lineo's real-time technologies, please visit www.lineo.com/products/realtime_linux/index.html.
>
> Lineo also makes hard real time available from our open-source site (opensource.lineo.com/projects.html). Users can download a full-version of RTAI (real-time application interface) and AtomicRTAI from this site.
>
> At LinuxWorld in San Jose this year, Lineo announced it would integrate real-time technology into the Embedix SDK, which Lineo will ship in Q4 of this year.

TimeSys Corporation:

A competitor of TimeSys recently announced that they are "the first to deliver hard real-time Linux". TimeSys would like to point out that this statement is false based on the following facts:

1. TimeSys has delivered to the market in May 2000, our TimeSys Linux/RT product that incorporates direct extensions to the Linux kernel that provides a strong platform for building hard real-time applications. These kernel extensions included support for guaranteed and deadline-aware CPU reservations with enforcement, 256 levels of fixed-priority scheduling, support for priority inheritance, support for periodic tasks, and high-resolution clocks and timers. These Linux kernel extensions, called the "Resource Kernel (RK)" have been downloadable from our web site (www.timesys.com/products) since May 2000. RK extensions are binary-compatible with Linux by definition and can actually allow Linux applications to be given real-time and QoS guarantees without accessing or modifying the application source code.
2. It is well known in the real-time systems community that fixed-priority scheduling combined with priority inheritance support and high-resolution timers is sufficient to build hard real-time systems. In fact, the most popular framework for building hard real-time systems is called RMA (Rate-Monotonic Analysis), which requires only these primitives. RMA is the ONLY framework supported by all major standards in the real-time systems marketplace, including Real-Time Extensions to POSIX, Real-Time Java, Real-Time CORBA, Real-Time UML, Ada 83 and Ada95. The competition does *not* support QoS guarantees, priority inheritance, high-resolution timers or periodic tasks.
3. In addition, to the direct Linux kernel extensions described in (1) above, TimeSys Linux/RT 1.0 and 1.1 also includes the RTAI layer from DIAPM, Italy. The RTAI layer is an independent higher-priority real-time kernel that runs below Linux. TimeSys Linux/RT includes both this higher-performance (but non-Linux-binary-compatible RTAI) layer *and* the Resource Kernel extensions. The two are mutually exclusive, but both support hard real-time applications.
4. TimeSys Linux/RT capabilities are not hidden or abstract; they can be explicitly visualized by the use of the TimeTrace product from TimeSys (please see www.timesys.com/products). The exact sequence of scheduling events and system calls occurring on multiple TimeSys Linux/RT targets can be viewed on a host and verified for strict correctness. In fact, code segments and system calls that take less than tens of nanoseconds can be measured *without* adding any code to an application.

Lineo:

Doc Searls (info@linuxjournal.com) is senior editor at *Linux Journal* and coauthor of *The Cluetrain Manifesto*.

Archive Index Issue Table of Contents

Advanced search

Using PostgreSQL

Reuven M. Lerner

Issue #79, November 2000

If you haven't already, perhaps it's time to see if PostgreSQL is the database for you.

When I first began to use Linux for basic Web development, I saw that my three primary tools—Perl, GNU Emacs and Apache—were already included. But, at least one thing was missing, namely a relational database. I had used databases (mostly Sybase) for about a year when I began working with Linux, and knew that I would need a good database server in order to create sophisticated web sites.

I had heard of PostgreSQL and, after learning a bit more about it, decided to install it. Unfortunately, my installation experience was less than pleasant, and I gave up after several hours. Another reason I gave up was that I found another database server, MySQL. MySQL did not have all of the features I wanted in a database, but it was close enough; I implemented many web applications for clients using it.

Things have changed quite a bit in the last five years: MySQL has been rereleased under the GNU General Public License and is slated to include basic support for transactions, while PostgreSQL is now remarkably easy to install and includes a wealth of features and functionality. Both are well known in the Free Software community as powerful programs that can help get your work done.

I still use MySQL for a variety of tasks and expect to continue doing so, but, increasingly, I find that PostgreSQL is a better fit for my needs. This month, we will look at PostgreSQL, starting with its basic features, to create a small web-based application that uses transactions. Along the way, I will try to compare it with MySQL, describing where one product might be better suited than the other.

Installing PostgreSQL

As of this writing, the latest version of PostgreSQL is 7.0.2, released in the spring of 2000. As with all open-source software, you can download the PostgreSQL source code from the Internet and compile it yourself. My office uses the Red Hat distribution, and I generally prefer to install software with RPMs for easier maintenance. We downloaded the RPMs from the PostgreSQL web site, installed them, and were up and running in almost no time.

Like all modern database systems, PostgreSQL contains a server that can handle connections from multiple clients. Typically, only one computer in a network is designated to be the database server, with the remaining computers configured as clients. (The server is often configured to be a client as well, in order to facilitate debugging and system configuration.) RPMs for the server typically begin with the name “postgresql-server”, while the client RPMs are named “postgresql” followed by the version number.

PostgreSQL clients exist—in source form and as RPMs—for most of the programming languages that people use in designing web applications, including Perl, Python, Java and PHP. If you intend to build any of these from scratch, you will need to install the PostgreSQL development libraries, either from source code or from the RPMs. I compiled Perl and its modules from the source code but, otherwise, took advantage of the RPMs and installed the precompiled binaries.

Using PostgreSQL

Like MySQL and many other relational databases, tables within PostgreSQL are grouped into a single “database”, much as files are grouped within a directory. PostgreSQL offers a great deal of flexibility when it comes to security configurations. Databases can allow or deny access based on IP address or user name, while individual tables and other objects allow various levels of access based on the user name. These configurations are performed in the pg_hba.conf file, which was installed by default in /var/lib/pgsql/data on my system.

By default, only one user, called “postgres”, is authorized to create new users or to create new databases. When you first start to use PostgreSQL, you will probably have to use **su** to change your identity to root, then su again to change your identity to “postgres”. Once you have become the postgres user, you can use the **createuser** program to create one or more new users. You can then indicate whether they are allowed to create new databases and/or new users.

The **psql** database client which comes with PostgreSQL is an excellent interactive tool for working with the database. Like the MySQL client program, it uses GNU ReadLine to provide Emacs-compatible keybindings for entering SQL queries directly. **psql** also provides an extensive number of help commands that begin with backslashes, such as **\h** (which displays help for any SQL command), **\d** (which lists available objects and details about those objects), and **\l** (which lists available databases). I use psql constantly to double-check that program-generated queries worked correctly; it is easy and quick to use.

PostgreSQL implements a large portion of standard SQL and, therefore, is easy to learn if you have previously worked with a relational database. We can create a simple table, as shown in Listing 1.

<u>Listing 1</u>

The **serial** data type is similar to MySQL's AUTO_INCREMENT tag. It provides us with a unique number each time we **insert** a new row into the table. **serial** columns use a PostgreSQL data type called a “sequence” on which the PostgreSQL **nextval** and **currval** functions operate. It is also possible to use a sequence directly (see Listing 2).

<u>Listing 2</u>

PostgreSQL distinguishes between single and double quotes, so be sure to say **nextval(`people_id')** and not **nextval("people_id")**.

Inside of psql, the semicolon is a synonym for **\g**, meaning “go and execute this query”. Outside of psql, it has no meaning and may even cause an error in your database driver.

PostgreSQL, unlike MySQL, is case insensitive when it comes to table and column names. However, I prefer to follow Joe Celko's capitalization rules for SQL: keywords in ALL CAPS, table names in LeadingCaps, and column names in all_lowercase_with_underscores.

As in MySQL, PostgreSQL allows us to designate which column is the primary key. PostgreSQL also allows for **unique** columns (and combinations of columns), as well as the creation of indices with the **create index** statement.

PostgreSQL's data types are slightly different from those in MySQL, but relatively easy to understand if you have worked with another database. **varchar** and **numeric** types are supported. Perhaps the most confusing difference between data types in MySQL and PostgreSQL is **timestamp**. Under MySQL, a timestamp column is automatically set to the value of the latest **insert** or **update**. In PostgreSQL, a timestamp column simply contains a date/time value.

PostgreSQL supports many standard SQL functions and many of the extensions that MySQL contains. The difference, of course, is in how they are implemented. For example, MySQL has a **regexp** function similar **like** function. PostgreSQL, by contrast, has implemented regexp functionality with the **~** (tilde) operator, perhaps because of Perl's popularity.

PostgreSQL also makes it possible to create new datatypes with the **create type** function. Indeed, one of PostgreSQL's claims to fame is that it is a hybrid of object-oriented and relational databases. I have never needed to use this functional, but it does seem to be an intriguing and powerful feature.

Referential Integrity

So far, PostgreSQL does not seem to be very different from MySQL, except in some of its basic syntax. But, it is here that the two databases begin to diverge. PostgreSQL includes what are known as “referential integrity” checks, meaning that I can define certain values in a row cand be defined as illegal.

One of the first things that a database programmer learns is that **null** indicates that a column contains no value—not even false, the empty string or zero. We can forbid a column to contain “null” by declaring the column to be “not null”. This is perhaps the simplest integrity check, with the database ensuring that the column in question can never contain an illegal value. In our People table above, the name column is defined as not null, telling the database that every person must have a name.

Many times, however, this is not enough. For example, the People table includes an e-mail column. Modern e-mail addresses must contain an @ sign in order to be valid. Using referential integrity, we can ensure that any e-mail address added to the database contain an @ sign. In order to do this, we use PostgreSQL's Perl-like ~ operator, which matches a string with a regular expression:

```
   email    VARCHAR(50)    NOT NULL    CHECK (contact_email ~ '@')
```

Without an @, an entered (or updated) value will be considered illegal and generate an error code. Having the database flag such an error might seem frustrating, but it is certainly better than having a database with incorrect values. I often use such checks to ensure that columns are not null and to forbid the empty string or other illegal values. For example, it is possible to define the People table with referential integrity checks (see Listing 3).

Listing 3

If I try to insert a row which violates any of these checks, PostgreSQL will refuse to do so (see Listing 4).

Listing 4

So it seems that my value for address2 was invalid. Indeed, I tried to enter an empty string here, which is not allowed. Rather, I should have entered a null value (remember, null and the empty string are distinct values). Sure enough, replacing the empty string with null allows the query to succeed. But the database refused to allow us to corrupt it with invalid information and did not insert the new row.

Foreign Keys

Referential integrity also means that one table can reliably point to another with "foreign keys". Most tables have a primary key, meaning a column (or set of columns) that uniquely identifies each row. For example, the databases of the United States Social Security Administration uniquely identify each American citizen with a Social Security Number. This number means that you can change your name, address, phone number, bank accounts and job, but the same SSN will still refer to you. In the same way, a primary key allows us to continue pointing to a particular row in a table without having to depend on any of the values in that row.

For example, let's assume that we want to create an Appointments table containing three columns (see Listing 5).

Listing 5

The table in Listing 5 allows us to indicate when we are meeting with each person, ensuring that the notes column is either null or non-empty, and that only one appointment can take place at a time. The person_id column is supposed to contain the person_id from the People table. However, what stops me from entering a person_id of five or 50? How can I be sure that the value is valid?

The answer is that we can set person_id to be a "foreign key" of People, meaning that People.person_id can only contain a value that is contained in People.person_id. We can add this constraint with the REFERENCES keyword (see Listing 6).

Listing 6

Transactions

One of the chief complaints that PostgreSQL users level against MySQL is the lack of transactions. If you have only used MySQL in your development work, you may wonder why you need transactions and how they fit into a database environment.

The basic idea behind transactions is that they group multiple queries into a single logical query. If any of the queries in the transaction should fail, the database is "rolled back" to where things were before the transaction started.

Thanks to transactions, you can be sure that a transfer of money from one account to another will not accidentally leave you with too much or too little money, even if the power fails in the middle of the transaction. Until the transaction is finally "committed", the database pretends that none of it has happened.

The MySQL documentation (and authors and support people) has its own philosophy of transactions, including commit and rollback, which runs counter to the "ACID" test that is prevalent among current relational databases. While I disagree with some of their conclusions, there is no doubt that the lack of conventional transactions in MySQL has made it a flexible and fast database, one that is well suited to web sites that perform many selects and few inserts and updates.

PostgreSQL follows the standard model fairly closely, making it possible to perform transactions without locking tables (as in the MySQL paradigm). To begin a transaction, use the **being work** statement, and to end one use either the **commit** or **rollback** statement. Users of Perl's DBI or of Java's JDBC can instead use the commit and rollback methods associated with the database connection object. Both DBI and JDBC operate by default in AutoCommit mode, meaning that each query is implicitly placed within its own transaction. To put several queries inside of a single transaction, a program must turn off AutoCommit mode, perform the transaction, perform a commit or rollback, and then (usually) turn AutoCommit mode back on. For example, let's assume that we have a separate Salaries table that indicates every employee's salary (see Listing 7).

Listing 7

Notice how the above table ensures that each employee can get only one raise on a given day, by setting a unique restriction on the combination of an employee and day.

We could presumably keep this information in the People table. However, putting salary information in a separate table makes it easier to hide the information from prying eyes. It also means that we can pull up an employee's entire salary history with a **SELECT** statement, while keeping the tables normalized and storing information about each person only once.

Having two tables for a single employee raises some issues. Most significantly, we want to be sure that any employee added to the People table will also be added to the Salaries table (it would be rather embarrassing to have an employee without any salary). Adding a new employee should be one logical operation but will require two insert statements—one into People and the other into Salaries. What happens if the database dies in the middle of the second statement?

Listing 8 contains a simple command line program (which could easily be turned into a CGI program) that creates a new employee, first adding a new row into the People table, and then adding a corresponding row to the Salaries table. We retrieve the person_id of the newly inserted employee using PostgreSQL's **currval** function. We then use that value to **INSERT** a row into the Salaries table.

Listing 8

We ensure that the two operations occur within a single transaction by turning AutoCommit mode off (setting it to 0). Once this happens, we are responsible for performing a commit when we are done. Without a call to **$dbh->commit**, PostgreSQL will assume that we want to roll back both of the insert operations, pretending that they never happened. If the program dies in the middle—which will happen if any of the SQL queries fails, since we have activated RaiseError—none of the changes will occur.

To trap the error and display a message to the user, we can use the block version of Perl's **eval** function, demonstrated in Listing 9. This runs the code inside of the {}, exiting from it, and setting the special variable **$@** if anything goes wrong. This technique of using eval to trap errors gives us a basic form of exception handling and enables us to print out only the errors we want. If we were to activate either the PrintError attribute or the “use diagnostics” pragma, Perl would print out more than just our simple message, confusing the user.

Listing 9

Views, Procedures, Triggers, and Rules

We have now defined three tables: People, Appointments and Salaries. What happens if I want to create a table listing all appointments scheduled with people, along with their salary histories? (This would be useful to have before annual salary meetings.) The table in Listing 11 will perform such a request, listing appointments in chronological order.

Rather than having to create this query from scratch each time, we can create it as a “view”. Views are dynamically generated tables with names attached that can largely be treated as read-only tables. We can create a view of appointments and salary information, as shown in Listing 10.

Listing 10

Notice how there is almost no difference between the generic select statement and the select used to create a view. Indeed, the only change that we made was in dropping the order by clause, because PostgreSQL has not implemented this functionality as of version 7.0.2. So, we can list all appointments in chronological order (see Listing 11).

Listing 11

Views have a number of advantages over simple select statements:

- They force more of the processing to take place on the database server. Since database servers are typically high-end machines, and because they have ready access to the data, they end up doing more of the work. The client spends less time creating dynamically generated SQL statements.
- Views have their own permission structure. For example, the personnel department at a company, but no one else, should have access to salary information. With views, it is possible to hide information and allow particular users or IP addresses to access the base table, but keep the view open to the general public.
- Perhaps most importantly, views let us think at a higher level of abstraction than basic tables. Views can perform a variety of calculations and manipulations on the values from our table, just as a simple select can. If you know that you will have to multiply all of the values in a particular column by three, you can create a new view which automatically performs the calculation. You no longer need to perform the calculation in the select statement, nor in the program that retrieves the values.

Views cannot take variable arguments, meaning that you cannot create a view that sometimes retrieves user names beginning with “A” and sometimes retrieves those that begin with “B”. To perform such an operation, you will need to create a procedure. PostgreSQL supports a variety of programming languages in which procedures can be written, including Pl/PGsql (which is similar to Oracle's PL/SQL language), PL-perl (procedures written in Perl), and Pl-tcl (procedures written in Tcl).

Once you have created procedures, you can then create “triggers”. A trigger is a procedure automatically executed when something happens in the system. For example, you can use a trigger to ensure that when a user is deleted from the People table, rows in the Salaries and Appointments tables refer to that user are also deleted. Without this, it is possible for a row in Salaries or Appointments to refer to a person_id that no longer exists. Triggers can be activated whenever someone performs an insert, update, or delete on a table, and can operate either before or after the action occurs.

Finally, views are normally read-only objects, since they are simply aliases for select queries. However, PostgreSQL has a sophisticated rule system that makes it possible to rewrite queries that fit certain criteria. Using rules, you can intercept an insert, update or delete that is targeted at a view, and rewrite it as a series of operations on one or more tables. Thus, an insert into ApptAndSalaryView could not be rewritten, since the user's e-mail address (from People) does not appear. However, an update would certainly make sense, and could be rerouted to modify the People, Appointments or Salaries tables as necessary.

Limitations

While PostgreSQL is rapidly growing in popularity, it does have problems. The PostgreSQL development team is well aware of these problems and seems to be dealing with them quickly.

The most pressing issue is probably the 8KB limitation on each tuple, or database row. This means that no row can contain more than 8KB of data. This affects many parts of the database's operation, from providing only moderate support for BLOBs (binary large objects) to preventing developers from creating even moderate-sized tables.

Other issues, such as the combination of views with unions, bit me (and my clients) on a recent programming project. This, combined with the lack of outer joins, has meant a number of workarounds in some recent projects. It is possible to get around most or all of these at the application level, but I am anxiously awaiting the addition of these features.

Working with MySQL has spoiled me somewhat, since the number of built-in functions is large and allows me to create database applications without creating my own functions. PostgreSQL has fewer built-in functions but, as we saw earlier, does allow me to create any that I might like, in a variety of programming languages. However, the installation and use of these languages is poorly documented; while they might be very powerful, it takes a while to get started with them.

Despite some benchmarks that were recently released by a PostgreSQL consulting group, it's safe to say that PostgreSQL is slower than MySQL. At the same time, I was pleasantly surprised to discover that the speed difference was not as great as I expected. Of course, this speed difference exists because PostgreSQL includes transactions and referential integrity, both of which require more processing and record keeping than MySQL's table-level locks.

A final drawback to PostgreSQL is that not as many web-hosting services offer it. This may be unimportant when working with dedicated servers, but some of my clients have the budget to only rent a virtual server. Database capabilities should be a consideration when looking for a web server, but, in my experience, developers often have less say in the tools used for a project than they might like. Still, if you are interested in something closer to a commercial database, including transactions and integrity constraints, PostgreSQL is your best choice.

Conclusions

While I continue to use MySQL, I am increasingly impressed by PostgreSQL and have begun to use it for a number of consulting projects. I have been impressed by its speed and versatility, as well as its future direction. The improvements between version 6.*x* and 7.*x* were staggering, and I look forward to seeing more!

If you are implementing database applications—including web/database applications—under Linux, I encourage you to take a look at PostgreSQL. Even if you decide to stick with MySQL, it is useful to understand how other databases work and why there is such a fuss about transactions, stored procedures and integrity constraints in the community of database programmers. And who knows? Maybe you will also find that PostgreSQL is better-suited to your applications than you think. One of the beautiful things about free software is that you can choose the tools that are best for your needs, and learning about PostgreSQL is a great step in that direction.

Resources

Reuven M. Lerner owns and manages a small consulting firm specializing in web/database application development. As you read this, he should be finished with Core Perl, to be published by Prentice-Hall. You can reach him at reuven@lerner.co.il or at the ATF home page, http://www.lerner.co.il/atf/.

Advanced search

Advanced search

Aging Systems for Flavor

Marcel Gagné

Issue #79, November 2000

Disclaimer: Absurd as it seems, the IDSA claims you can't do this. [--ED]

You see, François, the wonder of cooking with Linux. This tiny diskette contains a complete Linux distribution. With it, I can set up an IP firewall and gateway using nothing but an old 386. *Mais oui*, I read about it on the *Linux Journal* web site in their System Administration section. Ah, François, sometimes I long for the days of small, fast applications, requiring just the slightest hint of memory, the faintest glimmer of disk space, and nothing but a subtle *je ne sais quoi* to achieve grandeur. Cooking with Linux, all this is still possible. Of course, in the old days, it was not all productivity, *non*?

Oui, François. *Qu'est-ce que tu dit? Ah, mes amis!* Forgive me. I was reminiscing and did not notice your arrival. Please, come in. François will see you to your table, and I will choose a wine. Hm...what to serve?

Every once in a while, one longs for the days of some old hardware platform on which they first practiced their computing (or gaming) skills; a Commodore 64, a PET, an ATARI, an Amiga, a TRS-80 or even an IBM 1130. This is part of the normal human condition, something called “nostalgia”. Not unlike a fine wine, some of these old platforms seem to mature and take on subtle, elusive qualities that can best be appreciated in a relaxed, open atmosphere.

This month, in a special hardware issue, we are going to be cooking with Linux, but cooking up something decidedly “un-Linux” in the process.

Speaking of wine...François! Kindly bring up the '89 Pomerol from the cellar. Our guests are thirsty.

Emulators are quite common in the Linux world. You can run old DOS applications with **DOSEMU** and Windows applications with **WINE** (knowing full well, of course, that Wine Is Not an Emulator). To truly emulate something, you create virtual machines that run on your current machine. With Windows, this is done using products like VMWare (http://www.vmware.com/), which let you run full versions of Windows on Linux. Emulators can also bring back systems long ago thought dead.

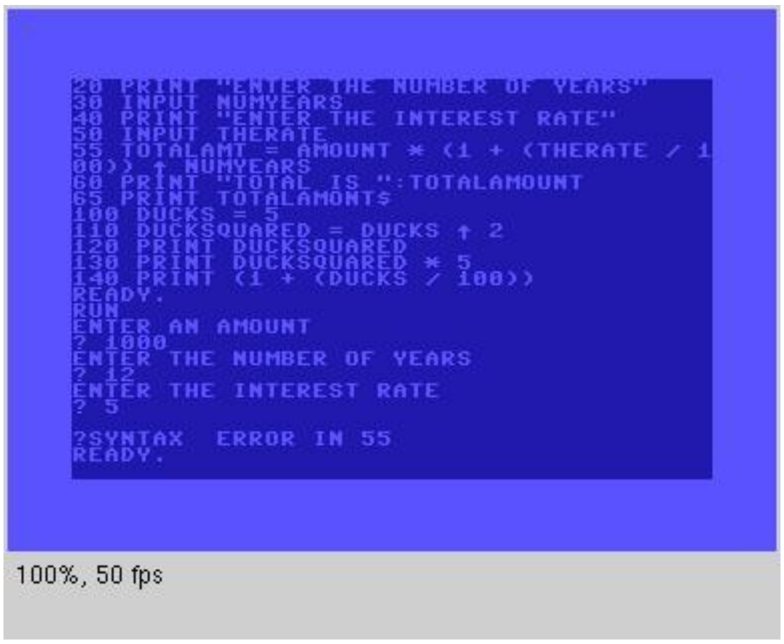
In the case of the products I describe here, you must remember that the material is often still copyrighted. Some emulators require that you have the original disks containing the OS or the ROM image. Making software backups of old ROM cartridges is beyond the scope of this article and would require that I keep François working far too late. However, emulator sites do store this type of information for various platforms. There also are a number of public domain programs designed for each of these platforms, so you can have the joy of playing a game with 8 bit graphics without the guilt. I will also hold off until the end of the article to give the appropriate URLs for each emulator I discuss.

Getting these emulators is as simple as visiting the appropriate sites on the Web. I won't spend a lot of time explaining how to go about compiling each one. The format is essentially the same for all.

```
   tar -xzvf emulator-1.01.tar.gz<\n>
   cd emulator-1.01
   ./configure
   make
   make install
```

First on the menu is an old favorite of mine. Once upon a time, your humble chef had a Commodore 64 (not to mention a PET and even a VIC20). For those who may still have old programs floating about and long for the days of your favorite game, complete with sound courtesy of Commodore's SID chip, try a little VICE, which stands for “Versatile Commodore Emulator”; it is indeed versatile. It can emulate the C64, the C128, the VIC20 and most of the PET models. The “VICE Team” distributes the program under the GPL.

Of all the Commodore models, I admit to having been quite attached to my old C64. On a whim, I decided to try a little basic program, for old time's sake. As you can see from the screen capture (see Figure 1), Marcel's Commodore BASIC is not as sharp as it once was. To launch a Commodore session, simply execute the following:



```
20 PRINT "ENTER THE NUMBER OF YEARS"
30 INPUT NUMYEARS
40 PRINT "ENTER THE INTEREST RATE"
50 INPUT THERATE
55 TOTALAMT = AMOUNT * (1 + (THERATE / 1
00)) ↑ NUMYEARS
60 PRINT "TOTAL IS ":TOTALAMOUNT
65 PRINT TOTALAMONT$
100 DUCKS = 5
110 DUCKSQUARED = DUCKS ↑ 2
120 PRINT DUCKSQUARED
130 PRINT DUCKSQUARED * 5
140 PRINT (1 + (DUCKS / 100))
READY.
RUN
ENTER AN AMOUNT
? 1000
ENTER THE NUMBER OF YEARS
? 12
ENTER THE INTEREST RATE
? 5

?SYNTAX  ERROR IN 55
READY.
```

100%, 50 fps

Figure 1. Marcel's Commmodore BASIC Is a Bit Rusty

```
x64 (to launch a C-64 session)<\n>
xpet (to start your very own Pet)
x128 (starts a commodore 128)
```

Et oui, there are more. The C64 was an actual computer, in that you had a keyboard where you could type and do work. Not all the machines of the late 1970s and early 1980s were designed for work. One of the most popular games of the time was the Atari 2600 Video Computer System; it seemed everybody but your chef had one, although he did visit friends who were willing to share. For those longing to relive those days, there is **Stella**, created by Bradford W. Mott (although many others share in its continued development and ports). Stella is distributed free of charge, but the license is not GPL. Once again, there are numerous freeware games and demos out there for the curious. Running a game with Stella is very simple:

```
xstella path_to_game
```

Figure 2. Ah, Those Lovely 8-bit Graphics

The Atari 2600 was only one of many systems put out by the company over the years. Petr Stehlik's first computer was an Atari 800XL, which he describes as something like a first girlfriend, somebody (or something, as in the case of the 800) that you will never forget. He is the current maintainer of the Atari 800 emulator, originally written by David Firth). Along with the help of others, he continues to develop the product.

There is a huge repository of Atari 8-bit programs sitting over at the University of Michigan software archives. A visit there will provide you with hours of nostalgic fun. In the arcade games section, you may even find the answer to that old question about the chicken and the road with a familiar game called *chicken*. A number of these programs are stored in the old **ARC** compression format. You may need to get something like **unstuff** (the Alladin Expander) to extract the files after you download them.

Another computer I was quite fond of in my early days, and my first exposure to assembler programming, was a TRS-80 Model 1, Level II. It was on this machine that I first learned BASIC (Fortran actually came first on an IBM 1130). For the Radio Shack TRS-80 enthusiast, you should find Tim Mann's TRS-80 pages with emulators, links to documentation and other TRS-80 sites. In fact, there is a wealth of information about every aspect of this nearly forgotten platform. This is a man who, at one time, was a TRS-80 systems programmer and has never gotten it out of his system.

Hardware emulators even make sense for people developing software for existing platforms. **Xcopilot** is such a program for your PalmPilot. Using the **getrom** program from the pilot-link utilities on your system, you can then use this X Window version of the **copilot** program to develop software or debug applications without risking your own Palm Pilot's sanity (I have crashed mine with experimental programs). Or, you can simply run the applications on your desktop if you happen to be fond of that Pilot interface.

Finally, for those with far too much time on their hands, I discovered the most unlikely emulator. Remember those annoying little electro-pets on a keychain, the Tamagotchi? If you miss your little friend while you are busy working, you could run **Ktamaga**, a Tamagotchi emulator for the K Desktop Environment (KDE).

If you find yourself looking for emulators that will run under Linux that I have not mentioned here, consider visiting Freshmeat.net. Doing a search using “emulator” will give you plenty to keep you busy.

Once again, it is time to close the doors here at Chez Marcel. Apparently, I will be the one to refill your glasses one more time before you go. It seems that François has disappeared. He is probably playing that old chicken-road-crossing game, *non*? Our François is a hardworking man, *non*? I hope this little tour has given you a taste of how your Linux system can help bring back those long lost friends of your early computing days. Until next time, your table will be waiting here at Chez Marcel.

Á votre santé! Bon appétit!

Resources

Marcel Gagné lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy, and edits TransVersions, a science fiction, fantasy and horror magazine (soon to be an anthology). He loves Linux and all flavors of UNIX and will even admit it in public. In fact, he is currently working on *Linux System Administration: A User's Guide*, coming soon from Addison Wesley Longman. He can be reached via e-mail at mggagne@salmar.com, and you can discover lots of other things from his web site, www.salmar.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Where to Install My Products on Linux?

George Kraft IV

Issue #79, November 2000

The details on proper application location.

For some software developers and Independent Software Vendors (ISVs), there are differing ideas about where to install one's applications and software packages. Some prefer to install in /usr/bin/ or /usr/local/bin/, yet others prefer the /opt/ directory. Your preferences may vary depending on whether you have a UNIX System V, Berkeley Software Distribution (BSD), or GNU/Linux background.

The Filesystem Hierarchy Standard 1 (FHS), Version 2.1, was written to eliminate these differences by specifying detailed guidelines as to where system services, configurations and software should be located on a UNIX or UNIX-based operating system. In detail, the FHS explains the content and purpose for each of the primary directories (see Figure 1).

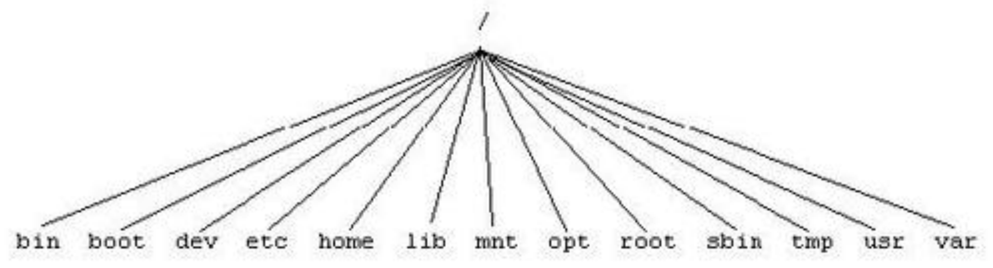


Figure 1. Schema of Primary Directories in FHS

In a nutshell, the base operating system's, or the distribution's applications are to be installed in /sbin/, /bin/, and /usr/. The system administrator can build packages from source and install them into the /usr/local/bin/ directory. However, the binary-only packages of nonessential applications and add-on software products should be installed in /opt/<package>/ directories, where <package> is the name that describes a software suite. The binary executables should be located in their respective /opt/<package>/bin/ subdirectory. If there are any accompanying UNIX manual pages, they should be installed in the /opt/<package>/man/ sub-directories.

The System V Application Binary Interface [AT&T 1990], the Intel Binary Compatibility Standard V.2 (iBCS2), the Common Operating System Environment (COSE), the Linux Standard Base (LSB), and the UNIX community in general have already established the /opt/ directory for add-on software.

The system administrator should create a separate disk partition for the /opt/ file system, and endusers should add /opt/<package>/bin/ and /opt/bin/ to their PATH environment variable. Usually the end-user's shell will find applications in their respective /opt/<package>/bin/ directories; however, the system administrator may have created symbolic links or wrapper scripts in /opt/bin/ for each package.

Host specific configurations for /opt/ binary executables should go in /etc/opt/<package>/ directories. These are the proper locations for configuration files of the /opt/ packages, because /etc/ is where all host specific system configurations reside on a UNIX-based operating system.

Variable files in a package, or files that change during the normal course of system runtime, should be kept in the /var/opt/<package>/ directories. The contents of /var/ are host specific and the directory is usually configured in its own file system to prevent the accidental filling of the root file system.

An exception to these rules is when it is necessary, or makes sense, for a package to install or create files elsewhere. For example, if a package were to create a new device, then it would be created in the /dev/ directory.

Now that we know the rules of the Filesystem Hierarchy Standard for add-on software packages, let's try to package and install a fictional software suite called Whizbang. If we are to follow the LSB specification, we should use the RPM Package Manager2 (RPM) and try to package our software for the /opt/ directory. This is shown at lines 18-20 of the whizbang-1.2-3.spec configuration file (see Listing 1). Line 8 shows how to make it relocatable so that the system administrator can install it elsewhere if so desired. However, installing in a nonstandard directory is not advisable.

Listing 1

Let's build Whizbang's RPM package as shown below. Using the whizbang-1.2-3.spec as the input file, RPM can build and produce a source package file whizbang-1.2-3.src.rpm and a binary package file whizbang-1.2-3.i386.rpm.

The book *Maximum RPM*, by Edward Bailey, or the RPM web site, http://www.rpm.org/, are excellent resources for learning to create RPM packages. For now, don't worry about the details, but recognize that there is a guideline on where to install most everything. To create the whizbang-1.2-3.i386, do the following:

```
# rpm -ba /usr/src/redhat/SPECS/whizbang-1.2-3.spec
Processing files: whizbang
Finding provides...
Finding requires...
Prereqs: /bin/sh
Wrote: /usr/src/redhat/SRPMS/whizbang-1.2-3.src.rpm
Wrote: /usr/src/redhat/RPMS/i386/whizbang-1.2-3.i386.rpm
```

Using the whizbang-1.2-3.i386.rpm binary package we just created above, we can now install it onto the system as show below:

# rpm -i \/usr/src/redhat/RPMS/i386/whizbang-1.2-3.i386.rpm

Now that whizbang is installed in /opt/whiz/bin/, try to run it from the command line. Did your shell find it? Was /opt/whiz/bin/ in your PATH environment variable? What if we wanted to make it more convenient for the enduser by creating a symbolic link to /opt/whiz/bin/whizbang from /opt/bin/whizbang? This could be done during the post-install phase of the RPM installation, as shown here:

```
%post
P=$RPM_INSTALL_PREFIX
mkdir $P/bin > /dev/null 2>&1
ln -fs $P/whiz/bin/whizbang $P/bin/whizbang
mkdir $P/man/man1 > /dev/null 2>&1
ln -fs $P/whiz/man/whizbang.1 $P/man/man1/whizbang.1
# EOF
```

This “relocatable” post-install code would be added at line 20 in Listing 1; however, the RPM **%postun** uninstall solution to remove the symbolic links is left as an exercise for the reader.

Sometimes it is necessary or desirable to install a software suite somewhere other than for where it was originally packaged. Now let's uninstall the whizbang-1.2-3 RPM package, and reinstall it in an alternate location.

```
# rpm -e whizbang-1.2-3
# rpm -i --prefix /usr/local
# /usr/src/redhat/RPMS/i386/whizbang-1.2-3.i386.rpm
```

In summary, binary-only packages of nonessential applications and add-on software products should be installed in the /opt/<package>/bin/ directory. We've seen the basics on how to create a relocatable RPM package and how to build it, and we have demonstrated its flexibility by being able to override the default /opt/ destination and selecting an alternate location. Following the FHS standard is the first step in making your GNU/Linux application more LSB3 compliant.

George Kraft IV , aka “GK4”, works in IBM's Linux Technology Center where he is currently assigned to work with the Linux Standard Base. George has been programming on the BSD operating system since 1982, when he started his computer science and mathematics studies at Purdue University. He began using GNU/Linux in 1993 when he installed SLS on his Gateway 2000 4DX2/66 system.

Seeking Set-Top Nirvana

Linley Gwennap

Issue #79, November 2000

Many companies see the television set top as the next battleground for delivering digital content.

Many companies see the television set top as the next battleground for delivering digital content. These vendors want to provide a variety of functions, including digital television, video on demand, program guides, program recording, web access, e-mail, shopping and games. However, there is no clear picture of which combination of functions will succeed. Today, Linux is a minor player in this market but could become more popular, depending on how the market develops.

The key players on the set top are the cable and satellite TV providers. They are currently replacing their "dumb" decoder boxes with interactive set-top boxes (STBs) at a rate of one million units per month. Most of the initial deployment has been in Europe, but the United States is beginning to ramp as well. For example, Adelphia Communications plans to upgrade more than 25% of its 5.5 million subscribers by the end of 2001. These STBs provide shopping, e-mail, web access and other interactive services, along with standard TV programming. TV providers are eager to roll out these new units to reap an expected bonanza of new service revenues. Forrester Research estimates that interactive TV services will generate as much as nine billion dollars (US) in annual revenue by 2004.

To deliver these services, STBs rely on a variety of software customized to display video and web pages on a television set. Some use middleware, such as Liberate or OpenTV, running on top of a simple real-time operating system (RTOS). Others use complete STB operating systems like Microsoft TV (based on Windows CE) and PowerTV. Service providers can build their own applications on top of these (APIs) using Java or other popular languages.

With few exceptions, Linux is being passed over for use in these applications. Coollogic has developed a set-top box based on its Coollinux version of Linux, but this product has few adopters. Liberate and OpenTV could run on top of embedded Linux. However, many of Linux's key features—such as robust networking support and range of APIs—have already been duplicated in these middleware products, which have the bonus of being able to work around the shortcomings of a simpler RTOS. A generic Linux simply can't compete against the customized platforms available from companies targeting the rapidly growing STB market.

While not a prime choice for STBs, Linux does have a foothold in devices that might be called set-top companions. These are devices (such as digital video recorders (DVRs) and Internet access devices) that connect to the TV but are typically downstream of the decoder box. The biggest win for Linux has been the Tivo DVR, widely praised as a groundbreaking device. But Tivo sales remain well below one million units, mainly because of its high price ($299 US plus $10 per month).

Video game makers are also fighting for a place on the set top. Led by Sony PlayStation, more than 20 million video consoles were sold last year. Few of these devices deliver any functions other than gaming, but this is rapidly changing, as the Sega Dreamcast now offers web access and Sony's PlayStation 2 adds DVD playback as well. Microsoft's X-Box, due next year, will offer both of these features.

Single-function devices, such as Tivo and WebTV, will be the first losers on the set top. Just about everything that connects to the TV will soon have web access, and DVR functions will be easily added to any STB with a hard drive, such as the X-Box or Phillips' AOL-TV unit. By integrating DVR and web functions into other devices, vendors can deliver these functions at a much lower cost than in stand-alone products and greatly increase deployment. Furthermore, adding these features to other products reduces the number of boxes and wires connected to the TV. User interfaces for these multifunction devices, however, must be carefully designed to provide simple access to all functions.

The fiercest battle will be between interactive decoder boxes and video consoles. As long as broadcast TV is the primary use of the television set, the decoder box will be difficult to displace, since the cable and satellite companies use proprietary digital formats to deliver their content. On the other hand, video consoles have highly desired content that is also available only in proprietary formats.

In the end, most homes will have an all-in-one decoder box that supplies a variety of interactive functions. With the possible exception of DVR, which requires an expensive hard drive, these functions will have no up-front cost, just a monthly fee. Other homes may have a simple decoder box and use the video game console for web access and other advanced functions. Unfortunately, this scenario leaves little room for Linux. Several major vendors are well ahead in delivering TV-optimized operating systems and middleware. Video consoles all use proprietary operating systems optimized for gaming performance. Linux is great for a company like Tivo that needs to put together a new product quickly. In the long run, though, the only chance for Linux to succeed in this market is if Coollogic, or a similar company, can convince major STB vendors to adopt the platform.

Linley Gwennap (linleyg@linleygroup.com) is the founder and principal analyst of The Linley Group (http://www.linleygroup.com/), a technology analysis firm in Mt. View, California.

Embedded Linux at LinuxWorld—What a Difference a Year Makes!

Rick Lehrbaum

Issue #79, November 2000

One short year ago, “embedded Linux” barely existed, from a commercial perspective.

One short year ago, “embedded Linux” barely existed, from a commercial perspective. August's 150,000-square-foot LinuxWorld conference (San Jose, California) clearly demonstrated how far embedded Linux has come in just a dozen months. Better than one in ten of the more than 160 exhibitors rolled out new products and services aimed at embedded Linux developers and applications. In case you missed the show, here's a brief summary about each of the exhibitors who highlighted embedded Linux in one way or another.

Adomo previewed an easily expandable, easily administered Linux-based home information system that offers fast access to information along with the ability to communicate easily. The system consists of a Linux-based home server and a network of thin terminal devices that provide access to information, Internet-based services, and group oriented applications. (http://www.adomo.com/)

Agenda Computing previewed the Agenda VR3, a new Linux-based handheld PC that will be introduced to the market this fall at a base price of $149 (US). The device, which is based on an NEC VR4181 system-on-chip processor, features an open source operating system and will be supported by a web-based open application development community. (http://www.agendacomputing.com/)

Applied Data Systems showed a number of single-board computers (SBCs) that are oriented towards solving the embedded display challenges of companies developing a wide range of embedded system products. Demos included a new Linux-based on-board interactive computer system being installed on golf carts; ViewML, Century Software's new embedded Linux browser, running on the 4'' x 6'' ADS Graphics Client StrongARM based single-board computer (SBC); and a “sneak preview” of ADS' new 3'' x 4'' “Bitsy”, StrongARM-based SBC, running Linux of course. (http://www.flatpanels.com/)

Computer I/O unveiled a unique embedded software and services “middleware solution” for the communications industry. The software enables Microsoft Windows clients to communicate transparently with Linux-based embedded data acquisition servers, without concern for operating system, network compatibility, or protocols. (http://www.computerio.com/)

Emperor Systems Software showed several examples of Internet appliance applications running on top of their embedded Linux and JAVA software technologies, including: “TV-Linux”, a TV set-top box solution; and “Studio-Linux”, a broadband server solution for interactive television. (http://www.tvlinux.com/)

IBM showed a wide range of Linux-based systems, from massive million-dollar systems, to the incredibly tiny embedded Linux wristwatch. IBM also demonstrated some new Linux-based Thin Client systems, and also announced availability of its VisualAge Micro Edition Java Virtual Machine for embedded Linux systems. (http://www.ibm.com/linux/)

Indrema promoted its upcoming embedded Linux-based TV set-top gaming console and game software developer network. The company issued a press release announcing the launch of its collaborative open game development initiative, and also announced two key strategic relationships, one with Red Hat (“Red Hat and Indrema Establish Linux Distribution for Console Video Gaming and Home Entertainment”) and the other, with Linuxcare (“Linuxcare and Indrema Team to Provide Embedded Linux Support for Video Game Market”). (http://www.indrema.com/)

Infomatec showed several example Internet appliance applications, including a set-top box and a webpad, that demonstrated typical products based on the company's Linux kernel based Java Network Technology (JNT) operating system technology. (http://www.infomatec.com/)

Intel demonstrated embedded Linux running on a wide variety of board-level computers, including a small Taiwan-manufactured Pentium-based embedded single-board computer. (http://developer.intel.com/)

ISDCorp although ISDCorp was recently acquired by LynuxWorks, ISDCorp still had their own booth, where they showcased the broad array of Royal Linux support for ARM and MIPS processors. Demos showed Royal Linux running on the Intel SA-1100, Cirrus 7200; LinkUp 7200, TI R4000, Toshiba 3912, NEC VR4121, and a Vadem Clio handheld PC. (http://www.isdcorp.com/)

Lineo showed a large array of eye-catching demonstrations featuring the company's Embedix Linux operating system. These included: Embedix RealTime, with the Linux Trace Toolkit; the Embedix SDK, with Target Wizard; Embedix running on the new Motorola MBX2000 embedded Pentium II-based SBC; Embedix GUI, running on a TV set-top box; uClinux, running on a Coldfire 5307 board in a demo that played MP3s on a pair of speakers and also controlled five robotic servo motors; uClinux, running on the tiny (SIMM-sized) “uCsimm” Dragonball-based SBC with its video displayed on a full-sized VGA LCD; and Embedix Linux with Embedix GUI, running a simulated automated teller machine application on an Advantech touch-screen computer. Lineo handed out free bootable floppies containing Atomic RTAI, an open-source functional real-time Linux distribution that fits on a single floppy diskette, and also made several new product announcements including: Embedix Real-time, High Availability Cluster support, and support for Embedix development on Windows hosts. (http://www.lineo.com/)

LynuxWorks, in addition to all the Royal Linux demos at the ISDCorp booth, LynuxWorks showed BlueCat Linux running three demos in their own booth. BlueCat Linux cross-development was demonstrated, showing the capability to support multiple embedded processor families at once, with the targets Internet-attached and controlled remotely from one host (using Rlogin). A second demo illustrated compatibility between LynxOS (a proprietary POSIX-compliant RTOS) and BlueCat Linux: it consisted of a pair of Motorola's new MBX2000 EBX SBCs—one running BlueCat, and the other running LynxOS—and showed how the same Linux multimedia application ('GIMP' could run equivalently on either OS. A third setup demonstrated high availability and fail-over capabilities, in the form of a chess game: a browser on a PC provided graphical display of the chess board; the chess program itself ran as a server on a pair of redundant system controllers; when the first controller fails, the alternate one automatically takes over the server function and thereby provides uninterrupted operation. Pretty expensive chess game! (http://www.lynuxworks.com/)

Metro Link showed their ability to develop custom embedded Linux-based graphics display solutions and technologies. One such technology is Micro-X, a scalable X server implementation for embedded systems and consumer electronics. In addition, Metro Link disclosed several new products that are currently in development, including a home information management technology suite, multimedia tools, and 3D rendering software (derived from OpenGL). (http://www.metrolink.com/)

MontaVista beneath the shadow of huge, suspended renditions of the company's signature hard-hat-wearing penguin and Kerbango's Internet radio, was a dizzying array of demonstrations of Hard Hat Linux and its features.

IBM's VisualAge Micro Edition embedded JVM, riding on top of MontaVista's Hard Hat Linux was shown on an Embedded Planet “Linux Planet” demo systems, running various touch screen interfaces and games. Ziatech (recently acquired by Intel) was demonstrating their unique “Ketris” CompactPCI hot swap and CPU fail-over technology, running on Hard Hat Linux of course. The Microwindows embedded GUI and windowing system, along with the ViewML browser, were shown running on MIPS, StrongARM, and PowerPC processors. Hard Hat Linux and its new Cross-Development Tools and Integrated Development Environment were demonstrated on the tiny Intrinsyc CerfBoard (serving web pages), on the IBM's PowerPC 405GP “Walnut” evaluation board, on Bus-Tech's 1U configurable rackmount appliance, on the new Motorola MBX2000 EBX form-factor SBC, and on an Intel “PICA” (x86) platform. Kerbango's Internet Radio (with Hard Hat Linux inside) provided audio entertainment, streaming audio broadcasts from Real Audio servers around the world. Finally, if you happened to drop by the booth at the right moment, you were treated to a really cool demo: a Compaq iPAQ running Hard Hat Linux and the Microwindows GUI, complete with handwriting recognition. (http://www.mvista.com/)

Motorola Computer Group took the wraps off their new highly integrated yet compact single-board computer, the MBX2000, in the EBX (5.75'' x 8.0'') embedded computer form-factor. The MBX2000 is based on a low power version of Intel's Pentium II processor (in BGA1 chip packaging), and includes a display controller and Ethernet interface, along with a long list of additional I/O ports. On the software side, Motorola announced a partnership with Red Hat to support the high availability requirements of telecom applications. (www.motorola.com/linux)

Synergy Microsystems demonstrated Linux running on their PowerPC-based VME and CompactPCI CPU boards, and announced an agreement to offer TimeSys' Linux/RT on Synergy's products. (http://www.synergymicro.com/)

TimeSys three demonstrations acquainted booth visitors with the Linux/RT operating system associated tools. A demo of the quality of service (QoS) capabilities of Linux/RT showed what happens to streaming video from a USB camera when processes are added to a system—with, and without, QoS parameters enabled. A second demo showed the capabilities of three new TimeSys tools—TimeTrace, TimeWiz, and TimeWarp—including modeling system scheduler performance under varying “what if” conditions. In a third demo, a Motorola Pentium-based CompactPCI computer board was operated as an embedded target system, and ran Linux/RT out of flash memory. (http://www.timesys.com/)

Transvirtual Technologies you could hardly get near Transvirtual's booth! What was all the excitement about? Transvirtual was demonstrating its new PocketLinux OS, an open-source embedded Linux implementation that runs on PDAs like the Compaq iPAQ and VTech Helio. (http://www.transvirtual.com/)

Trolltech announced a graphical user interface (GUI) toolkit and windowing system for embedded Linux based devices, called Qt/Embedded. In its smallest configuration, Qt/Embedded loads from just 700K of ROM (disk). (http://www.trolltech.com/)

ZF Linux Devices showcased the company's MachZ X86 system-on-chip processor and associated development tools and platforms. One cool demo consisted of a very small self-contained computer board bearing the MachZ, several active components, and a small LCD display. Another demo showed the MachZ Integrated Development System; the system contains an ATX form-factor PC-compatible motherboard based on the MachZ system-on-chip processor, plus PCI plug-in cards for video and networking, and serves as a convenient evaluation and development environment. A third demo highlighted the new zPortPC Home Internet Appliance system—an aesthetically packaged device intended to serve as a small, cost-effective platform for a wide range of Internet appliance applications. Choose your favorite color from a rainbow of options! (http://www.zflinux.com/)

Well, that's all for now. See you at the next LinuxWorld Expo and Conference next January, in New York!

Rick Lehrbaum (rick@linuxdevices.com) created http://www.linuxDevices.com/ the “embedded Linux portal”, which recently became part of the ZDNet Linux Resource Center. Rick has worked in the field of embedded systems since 1979. He cofounded Ampro Computers, founded the PC/104 Consortium, and was instrumental in launching the Embedded Linux Consortium.

Archive Index Issue Table of Contents

Advanced search

Gadges and Gadgets!

Stan Kelly-Bootle

Issue #79, November 2000

Stan discusses gadgets.

It's strange how often *my* essential tools of trade are readily dismissed as gadgets (mandatory usage being “mere gadgets”). The doohickeys that *you* cherish are, of course, totally pathetic in my eyes, revealing you as a “sharper image” slave to self-defeating, fashionable mechanisms. Price and pride of ownership (closely related, nay, tightly bound) have clearly blinded you to the rational basics that guide my purchases.

The origin of “gadget” is, as the more honest lexicographers say, *orig. uncertain* or *etym. obscure*. Some guess at the French *gâchette*, but this was originally the safety lock-catch on a door or pistol, a reference to security and hard to reconcile with our modern “mere” gadgetry. Later, *gâchette* moved to become the pistol's “trigger” (as in *Elle a la gâchette facile*, “She's trigger happy”) which is perhaps more cognate with current gadget usage (see Note). She clicks to conquer, opening car/garage/home doors and TV channels, and, replacing old-fashioned scholarship, clicks to access the webbed windows of opportunity. Losing your clicker (I risk a gender-switch) is gadget-impotence, akin to Milton's blindness. Fear not: there's a meta-gadget you can click that will locate (beep-beep) your misplaced clicker. Lord knows what deeper levels of technology exist for recovering lost meta-gadgets. And now that mice, keyboards, modems and monitors (more) are becoming footprint-loose, wireless “remotes”, there's a growing chance of mislocation.

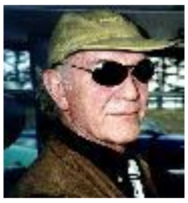
My own amateur spin is that “gadget” has morphologically diminutive implications (easy for me to say) that have added to its derogatory usages. I would back-form the term “gadge” to describe my own solidly justified possessions, regardless of price, conversation-party-piece, physical size and weight.

We are, patient readers, converging to a topic relevant to Linux, the OS we all love. I refer to the test known as *productivity*, an apparently simple econometric criterion that should (fat chance) dictate all our choices. The TV gadget heyday ads of the 1950s and '60s showed a trim USA super-efficient Mum clicking breakfast for hubby and her 2.7 adorable kids, then clicking the washing up. The general mantra “labor saving” (since applied to all forms of automation) left open the questions, “Saved at what cost?” and “time saved to do what else?” There's the ancient Time & Motion quip: Expert: “If you follow our plan, you could paint twice as many fences.” Tom, the painter: “But there aren't twice as many fences to paint.”

The equation has been obscured by the more subjective notion of “personal productivity”, yet even at the corporative level, after endlessly refined techniques and “studies at well-known West Coast Universities”, it remains dubious whether we can ever well-order our choices of computer languages, operating systems, development methodologies or guiding columnists.

Except to assert that my gadges are the best.

Note



Stan Kelly-Bootle (skb@atdial.net) has been computing on and off since his EDSAC I (Cambridge University, UK) days in the 1950s. He has commented on the unchanging DP scene in many columns (“More than the effin' Parthenon”--Meilir Page-Jones) and books, including *The Computer Contradictionary* (MIT Press) and *UNIX Complete* (Sybex). Stan writes monthly at http://www.sarcheck.com/ and http://www.unixreview.com/. Visit his home page at http://www.feniks.com/skb/.

Let Freedom Ping

Doc Searls

Issue #79, November 2000

Doc discusses business.

From the cowardice that shrinks from new truth,
From the laziness that is content with half-truths,
From the arrogance that thinks it knows all truth,
O God of Truth, deliver us.

—Source Unknown

I can't stand listening to Ralph Nader. His rap resonates with a lot of people, but for me the guy is strumming on a brick.

See, I love business. And I love markets, which are the social environments of business. They're fascinating to me, especially right now, when the sociology of business is changing so rapidly. Many of those changes start with the sociology of software—most notably the Free Software and Open Source movements. That makes business *really interesting* right now.

But Ralph hates business. Ever since he made consumer advocate a job title and became its permanent exemplar, he's been out to protect consumers from producers, no matter what.

Pick any big business and Ralph has a beef with its producers. Take air travel, for example. Since airline deregulation, a system built primarily to serve the world's upper business castes has been transformed into a bus system for everybody. As affordable transportation, a coach seat from New York to Chicago on American Airlines today is about the same price as it was on Greyhound in 1960.

Considering what it's asked to do—transport millions of people millions of miles at great altitudes at explosive speeds in fuel-engorged metal contraptions between airports built to handle far less traffic, every day, and do it while killing people at a rate far lower than automobiles—the airline industry delivers a service that routinizes the miraculous on a grand scale.

But, it has problems. There are lost bags, late arrivals, suspicious cancellations, bad food, stale air...all the stuff comedians have been joking about for generations. So comes the question: What do we do about all those problems?

Well, it depends on whether “we” are *the government* or *the market*. Ralph thinks *We the People* means *We the Government*. So do most of your basic Democrats. Your basic Republicans think *We the People* means *We the Market*. So Democrats like solving problems with Big Government while Republicans like solving problems with big market.

The problem is that neither side has ever seen a big market in which there has been much balance of power between supply and demand. Supply has pretty much controlled demand ever since Industry won the Industrial Revolution. That's why those of us who listened in history class remember a little something about James Watt's steam engine, James Hargreaves' spinning jenny and Eli Whitney's cotton gin, while we remember zip about the angry losers who broke into Hargreaves' home in 1768 and smashed twenty of his jennies. On the road of history, those guys were pavement.

We do remember something about the Luddites.

Here's some context from the French historian Elie Halevy, who wrote the six-volume *History of the English People in the Nineteenth Century* . In the first volume, originally published in 1913, he writes: “The equilibrium of society was overthrown to the detriment of the country districts, and to the advantage of the towns which were rapidly increasingly both in number and in size.”

Production, Halevy notes, changed from a local and artisan-based system to a centralized factory-based system, requiring workers and their families to abandon their rural homes and move to the cities and towns surrounding the new factories. This was no abstraction. It was the relocation and enslavement of a defeated people: “In these vast urban masses and in the manufacturing districts surrounding them the established social fabric was completely shattered.”

Luddites were craftspeople who refused to admit defeat. “A movement of insurrection on a large scale was organized in 1811,” Halevy writes:

> For two years the Luddites, as these revolted workmen were called, smashed (knitting) frames by the hundred, pillaged houses, and assaulted or killed obnoxious persons. The agitation spread to the neighboring districts and caused panic throughout the length and breadth of England. Cobbett extolled the Radicalism of Nottingham; Byron sang the praises of the Luddites.

So does *The Working Class Encyclopedia* (http://hammer.prohosting.com/~penz/encycl/), but with a different spin:

> The original Luddites operated in the Midland counties of England in 1811 and 1812. They did not roam the country, destroying any machine they found, as popular belief has it. They did their research and attacked only the looms belonging to scab Weavers who undercut the price of Handweavers. Often looms belonging to different Weavers were set up in the same room within a factory. The looms of fair-priced Weavers were left untouched.

Note the capitalization of the noun “Weaver”. This is a handy reminder to check your surname for the fossil remnant of an ancestor's role in a real marketplace. If it's Weaver, Weber, Tanner, Smith, Baker, Shoemaker, Shumacher, Merchant, Marchand, Hunter, Farmer or Fermier, your signature is a living reminder of the time when markets were profoundly social, when one's “living” and one's name were the same thing. Supply and Demand, Production and Consumption were all no farther apart than Vendor and Customer—the distance of one handshake.

The hearts of their cultures were the markets where they met and did business.

The beginning of the Industrial Revolution was the end of craft. While a few people continued to produce goods on their own, their economic significance was trivial. What mattered were the mines, factories, mills and transport systems that made and moved goods from production to consumption.

Markets were reconceived as commercial habitats where schools of fishlike creatures called consumers were organized by appetite and fed products pumped down distribution pipes that terminated in retail outlets. The culture this produced was often called “consumerism”, but the better label is producerism: an economic and cultural system where producers were in charge.

This is the vast economic machine that Ralph Nader—the ultimate Luddite—is out to break. So he attacks it everywhere producers make life hard for consumers, such as coach class in airplanes. “One would think that buying an airline ticket for a seat on the plane would include knees along with toes and torsos," Ralph writes. "But since airline deregulation over twenty years ago, passengers who do not fork up the dollars for a first class seat are treated more as cattle than as customers when it comes to space."

Well, not entirely. Airlines also had to find ways to handle demand for more passengers on a finite number of routes. Still, some airlines are starting to treat coach passengers as full-bodied bipeds again. American Airlines, Ralph observes, "will give them three to five extra inches of space in about a year." Then, he adds, "Hooray for small favors."

He sees no reason why big government shouldn't correct the airline industry. "Within the next 60 days the Aviation Consumers Action Project (ACAP) will file a petition with the FAA and the Department of Transportation to set minimum leg standards, unless the industry follows American Airline's lead in increasing coach class leg room to tolerable levels", Ralph writes. He also calls for a Six-Footer's Club, "to advance normal comforts for tall people". If you want to find out more, he provides a PO box in Washington, D.C., plus a phone number, but not a web site (although I found the quoted piece at http://www.nader.org/).

As a working (and consuming) class hero, Ralph is right up there with Karl Marx, Rosa Luxemburg and César Chavez. No other individual has had more influence on the biggest industry in the United States—automobiles—than Ralph Nader. The man has single-mouthedly saved thousands of lives. We owe him.

So why can't I listen to him?

Because he has already won and doesn't know it. The Industrial Age is over, and the Information Age is well underway. Workers are walking away from rusty old industrial machines that are dying for a single transcendent reason: they mistook people for parts. Now those people are going off to ply their crafts as competent human components of better, smaller machines—or large machines with better, smaller, more craftlike, i.e., autonomous—parts. These are machines they run, not machines that run them.

Consumers—those poor victims Ralph still fights for—are starting to discover what they really are, which is *customers*. If they don't like what they find in the market, it has never been easier for them to go make those things themselves, or to draw the attention of smart producers to the presence of demand. In this new age, the threshold of enterprise is so low it verges on zero. The thresholds of creation and innovation aren't much higher, which is why product and service choices spread wide everywhere supply hears demand. And what choice does supply have but to listen? If they don't, somebody with better ears will get the business.

The world is working like a real marketplace again, for a single reason: it's wired that way.

Who wired it? Try the Open Source development community, including the Free Software movement from which it grew. If you want to see a model of a craft-driven world in which every “worker” is a free, autonomous and effective agent, this is it. Show me a large technology producer (e.g., IBM, Dell and HP) that now proclaims itself as Linux-committed, and I'll show you a company in compliance with its software engineers as well as its customers' software engineers. All of those engineers like Linux and open-source software because it's easy to craft. It still might not be easy to use, but the software craftspeople are taking care of that. This was made especially clear by the GNOME, Helix Code and Eazel developers at LinuxWorld Expo in August 2000. Finally, we had an answer to the innovation vs. implementation question. These geeks can do both, thank you.

These are the same geeks Ralph addressed at The Bazaar last December, (his speech is at www.cptech.org/ms/harm.html). In that speech, he expressed extreme agreement with the government's case—and judgment—against Microsoft. I just checked on Altavista to see how many pages link to that speech. Try six. A total of 1,748 pages point to the Consumer Project on Technology (http://www.cptech.org/) domain, where the speech resides. Ralph's own home page (http://www.ralphnader.org/) makes an even sadder case, attracting a whopping total of five inbound links.

Perspective: Slashdot's number is 28,303. Which goes to show there aren't a whole lot of consumers left in the Open Source development world.

Or anywhere, which is Ralph's real problem. Ralph has a vested interest in the consumer, just like a Marxist has a vested interest in the worker. His heart is in the right place, but that place is being vacated by people who are starting to notice they have real power that power doesn't come from social and political movements, or from those movements' leaders. It comes from themselves.

Consumers and workers are rhetorical relics. The Net is uniting both, and they're throwing off their chains. Industrial communism and capitalism are both terminal. They can't survive in a networked marketplace, where “We the People” means exactly what it says.

We have real challenges in this marketplace, not the least of which is understanding how it's going to work, now that “the People” are working both sides of the supply-and-demand handshake. What are the new social contracts? What laws do we really need, and why are we keeping the ones we don't? How can we truly help those who can't help themselves? What are our agreements about privacy and anonymity, and how do we organize them? How do we build the needed infrastructure around our new Commons, and how do we keep those stuck with dead industrial market models from wasting our time?

We can find a democratic answer in the networked market's most honest voting mechanism. It's called the hyperlink. If what you say doesn't attract many, maybe there's no market for it.

Doc Searls (info@linuxjournal.com) is senior editor of *Linux Journal* and coauthor of *The Cluetrain Manifesto*. His opinions are his alone and do not express those of *Linux Journal* or SSC.

Archive Index Issue Table of Contents

Advanced search

MySQLGUI—the MySQL Graphical Client

Bill W. Cunningham

Issue #79, November 2000

It's free, easy to install and works like a champ.

- Manufacturer: TcX DataKonsultAB
- E-mail: sales@mysql.com
- URL: http://www.mysql.com/
- Price: Free (under the GNU GPL)
- Reviewer: Bill W. Cunningham

It's rare to find a Linux enthusiast these days who hasn't heard of the MySQL relational database server from TcX DataKonsultAB, of 0. It's free, easy to install, works like a champ, and has had excellent treatment by Reuven Lerner in his column, At the Forge (see *Linux Journal*, August 2000).

Many first-time MySQL users, upon reading the MySQL manual which comes with the source code, have undoubtedly balked at the manual's instructions on creating a new user:

```
VALUES('localhost'bill'PASSWORD('semperfi'n>
      'Y'Y'Y'Y'Y'Y'Y'Y'Y'Y'Y'Y'Y'Y'
```

The SQL “create user” and “grant...” commands work with MySQL too, but they require typing practically as meticulous as the above. At this point, some of us have lamented the lack of a graphical front end to all this. Something like Oracle's **svrmgrm** utility would sure be nice.

Happily, there is a graphical client for MySQL, and it's available from the same people who make MySQL. It's called mysqlgui. There are some other graphical interfaces to MySQL, for example, xmysql and xmysqladmin, but MySQLGUI is unique in that it performs both client and administrative operations in a single package.

None of the books currently available on MySQL mention MySQLGUI at all. Even the MySQL.com web site does not give it any attention. I came upon it by chance at the MySQL web site and decided to download it and give it a try.

MySQLGUI is available in three formats: source code, static binary and semi-static binary. I got the static binary, and it has worked fine. Although the accompanying documentation stresses that MySQLGUI is beta software, I haven't managed to crash it yet.

MySQLGUI is the work of Sinisia Milivojevic, of Larnaka, Cyprus. He reports that MySQLGUI actually has a large base of users worldwide. Its status as beta software notwithstanding, MySQLGUI is very stable—almost devoid of known bugs. It is built with the fltk library. TcX are planning a new GUI client within the coming year, one based on the QT or GTK libraries instead. The basic look may change somewhat in the new version, but all functionality will be retained.

Although the documentation on MySQLGUI is a bit spartan, its built-in functionality is rich indeed. You have to get it up and running and experiment with it to appreciate it. It quickly becomes apparent that a great deal of thought and planning went into MySQLGUI's development. Here is a brief synopsis of MySQLGUI's features.

Assuming you have MySQL server installed and running, you need only download the MySQLGUI source or binary and run mysqlgui from an xterm.

The MySQL Client greeting displays, followed by a user password dialog. After entering the user's password (if any), the MySQL Client main dialog takes over. It is the master control panel from which all other tools are launched. It also is where you can connect to the server, select a database, and run interactive queries against one or more tables. Figure 1 shows the main dialog with a user connected, a table selected, a query entered and the query results displayed. Clicking on the column headings in the query results window will sort the results based on that column's values. The first click sorts ascending, a second click sorts descending, and a third click unsorts. Saving the optionally sorted results to a file is a simple click away (see Figure 1). This ad hoc sorting functionality saves many keystrokes when compared to the MySQL command line client. Any SQL command can be executed in the query input field—not just queries. And the commands entered are saved in the lower command history area, where they may be recalled and re-executed later (even in subsequent sessions).

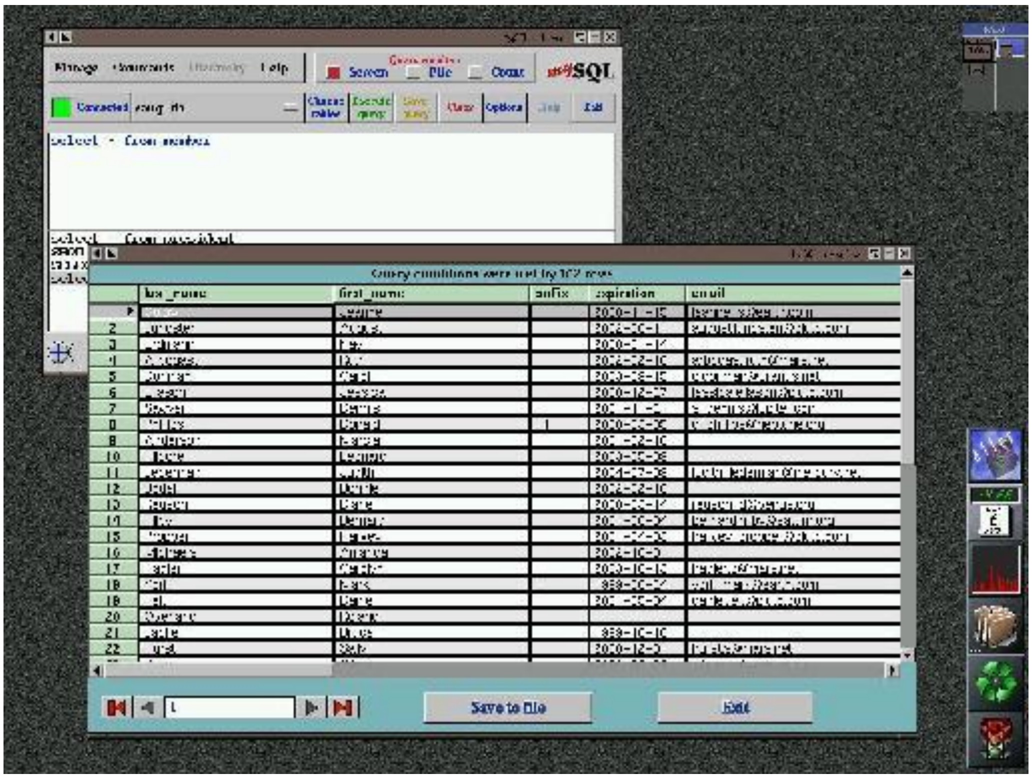


Figure 1. MySQL Client Main Dialog

MySQLGUI really shines when working with users. For this purpose, it has the Grant/Revoke tool. Figure 2 shows us about to create a new user, “annette”. She will have select, insert, update, and delete privileges on the “cigars” database on localhost. This is much easier, faster and more intuitive than either of the command-line procedures above. One thing perhaps not obvious is that we use the Grant/Revoke tool to create a new user. There is no Create User tool. However, doing it this way allows you to create a user and grant her privileges all in one shot. MySQLGUI, like the MySQL server, was designed to be fast, not fancy. Similar tools enable creating/dropping databases, tables and indices.

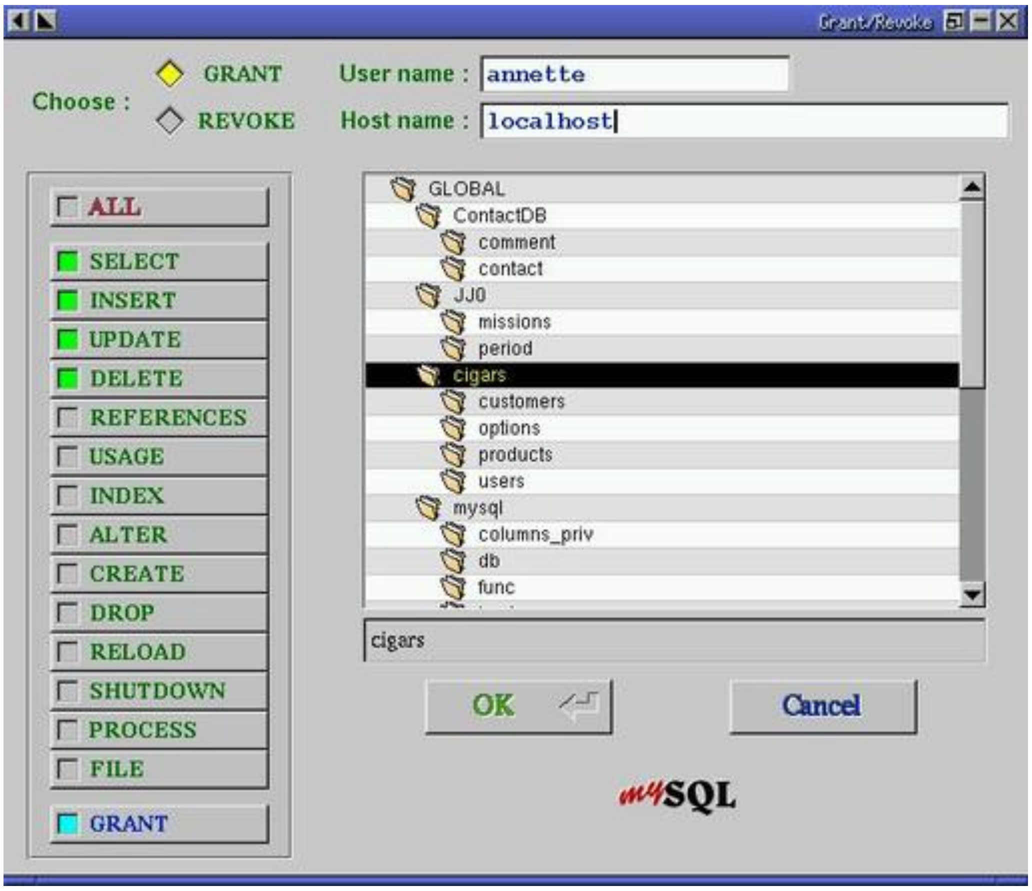


Figure 2. Creating a New User

Another useful tool is Table 0. Given time, you can accumulate a lot of databases and tables. Tables often have similar names. Figure 3 shows the database and table 0 tool. It allows you to browse all the databases and tables on your computer. This is quite useful, and I tend to keep this window open all the time when working with MySQLGUI.

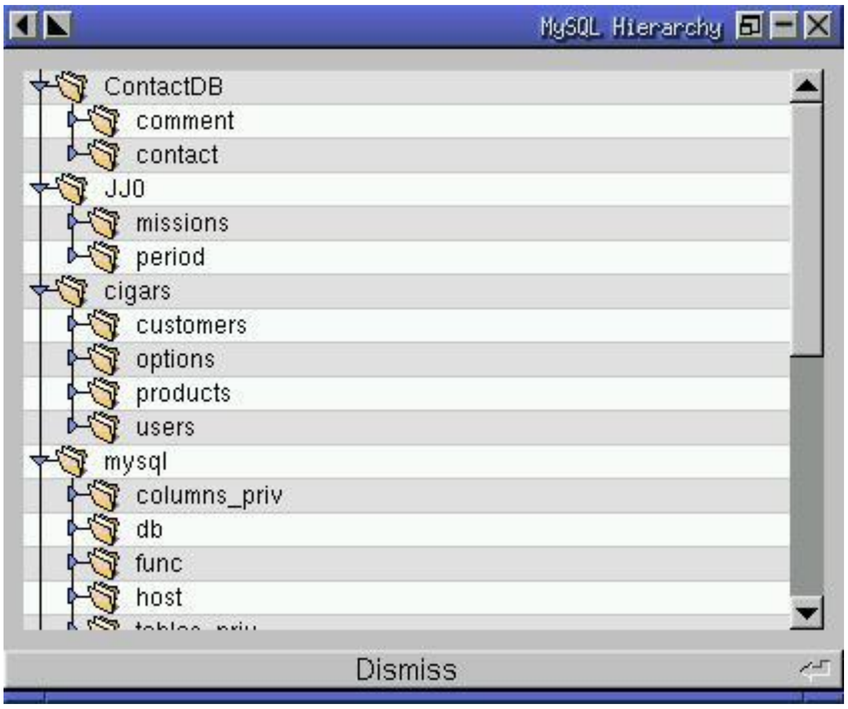


Figure 3. Table 0 Tool

I have touched on just a few of MySQLGUI's main features here. Suffice to say that it can perform any operation that one can perform with the command-line client. For simply issuing queries and sorting through your data, it may be all you need. And if you're a developer, it can cut down time spent formulating queries and administering your system. MySQLGUI is a first-class piece of work.

Resources

Bill Cunningham (cunninghamb@cherrypt.usmc.mil) is a systems administrator in Charlotte, NC. Recently retired from the US Marine Corps, he is active in the local Linux community and enjoys running, camping, family activities, and of course Linux!

Archive Index Issue Table of Contents

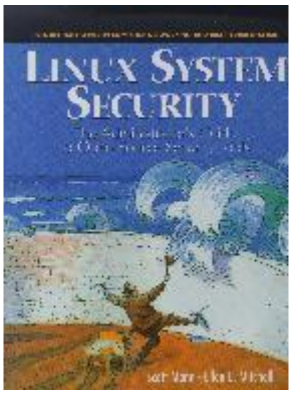
Advanced search

Linux System Security: The Administrator's Guide to Open Source

Ibrahim Haddad

Issue #79, November 2000

This book is intended to provide readers with skills, knowledge and tools that will allow them to prepare their systems for use in production environments.



- Authors: Scott Mann and Ellen L. Mitchell
- Publisher: Prentice Hall
- URL: http://www.phptr.com/
- ISBN: 0-1301-5807-0
- Price: $48.99 US
- Reviewer: Ibrahim F. Haddad

Linux System Security offers ways to protect Linux systems from break-in, as well as to detect evidence of attacks quickly. The book is intended to provide readers with skills, knowledge and tools that will allow them to prepare their systems for use in production environments. The methods discussed are from the perspective of restricting use to authorized access and making it as difficult as possible for crackers to gain access.

The book covers all aspects of Linux security and has plenty of practical tools and techniques for achieving it. The authors discuss common hacks and penetrations of Linux systems and show administrators how to protect themselves, set traps and trail hackers, using publicly available, open-source security tools. The tools are used to analyze, protect and monitor systems and networks.

In order to provide an accurate representation of the book's contents, the following is a summary of each of the 18 chapters in *Linux System Security*.

- Chapter 1--The authors guide the reader through a system vulnerability survey and discuss security policies. Various types of vulnerabilities and attacks are outlined, which is handy for people with no previous exposure to these issues.
- Chapter 2--A good overview of how to prepare a Security Policy and a useful framework for its implementation.
- Chapter 3--Background information on BIOS passwords, LILO, startup scripts, TCP/IP networking and cryptography is offered. Concepts and utilities are presented that are referred to throughout the book.
- Chapter 4--Necessary basic security issues related to user and group accounts management, using the root account, files and directories' permissions as well as file system restrictions are discussed.
- Chapter 5--Thoroughly pluggable authentication modules are presented along with a practical and comprehensive overview of PAM, its configuration and administration.
- Chapter 6--An in-depth discussion is offered of two different one-time password programs, S/Key and OPIE, and how they reduce considerably the risks associated with system access by utilizing a password only once.
- Chapter 7--System and connection accounting are explained. It describes in detail the commands that allow information collected by the accounting system to be viewed.
- Chapter 8--The **syslog** (system logging) utility is covered in great depth; syslog, its workings and the /etc/syslog.conf configuration file are all discussed. This chapter is the most informative piece on syslog I have ever seen.
- Chapter 9--An explanation of how to obtain, install and configure the Superuser utility, it talks about **sudo's** options, features and vulnerabilities.
- Chapter 10--The features, functionality and weaknesses of **inetd**, **TCP_wrappers**, the **portmapper** and **xinetd** are covered.
- Chapter 11--Implementation and configuration of the secure shell, SSH, one of the most important utilities in the public domain, is explained. The

authors describe how to build an encrypted tunnel between two or more hosts, protecting all aspects of the communication.

- Chapter 12--Crack, a tool that attempts to guess passwords, receives an in-depth explanation of how to build, configure and use it. The authors did not fail to address the ethical issues surrounding such a tool.
- Chapter 13--How to audit the system with Tiger, a set of scripts and programs that help identify system vulnerabilities is explained. The authors provide an overview of, how to obtain, install, configure and use it.
- Chapter 14--An overview of Tripwire, which acts as a valuable alarm system. The authors describe how to get, install and configure it, as well as how to securely store its databases and configuration files. Any Tripwire user will find this chapter valuable for its explanations and information.
- Chapter 15--Two publicly available tools to protect data through encryption are explored and compared. The Cryptographic and Transparent Cryptographic Filesystems (CFS and TCFS) that assist the system administrator secure data.
- Chapter 16--The focus is on packet filtering with the **ipchains** utility, and how to configure this utility to limit connections through a Linux system connected to two different networks.
- Chapter 17--Log file management as an essential part of system security and various log management tools, such as **logrotate** and **swatch**, are discussed.
- Chapter 18--An overview of the book's topics is offered along with ways to simplify the process of implementing, configuring and utilizing Linux security features and various publicly available tools.

At the end of the book, there are two appendices. Appendix A provides a list of web sites, e-mail lists and news groups that offer additional information about securing computer systems. Appendix B provides a list of several other tools that were not covered in the book.

Linux Systems Security is an essential book for system administrators and security professionals. It covers topics related to Linux systems security with a focus on freely available tools. The book helps identify system vulnerabilities and offers plans for security administration. It highlights how to detect intrusions and how to secure file systems, e-mail, web servers and other key applications. The book also emphasizes administrative security duties with discussions of system accounts, logging, superuser safety and secure network services.

A nice feature of the book is that the authors approach the subject from a practical point of view by emphasizing the use of software and providing references at the end of each chapter for further investigation. Another characteristic is the use of many examples, charts, tables and graphs to illustrate complex processes and concepts.

If you depend on Linux to run mission-critical networks, and you want to protect your Linux system, the procedures outlined in this book will certainly reduce your system's level of vulnerability.

Ibrahim F. Haddad (ibrahim.haddad@lmc.ericsson.se) works for Ericsson Research Canada in the Systems Research Division. He is currently a Dr Sc candidate in computer science at Concordia University in Montréal.

Archive Index Issue Table of Contents

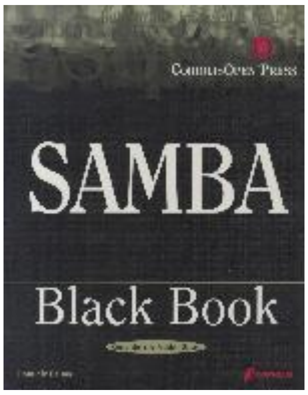
Advanced search

SAMBA Black Book with CD-ROM

Daniel Lazenby

Issue #79, November 2000

SAMBA Black Book is about providing server message blocks (SMB) networking services in a mixed operating system network.



- Author: Dominic Baines
- Publisher: CoriolisOpen Press
- URL: http://www.coriolis.com/
- Price: $49.99 US
- ISBN: 1-57610-455-9
- Reviewer: Daniel Lazenby

SAMBA Black Book is about providing server message blocks (SMB) networking services in a mixed operating system network. This 13-chapter book offers the reader foundational information, advanced topics and explores administrative issues. The book is designed and written to present the steps involved with building Samba servers. The book's examples focus on running SMB services on top of NetBIOS over a TCP/IP network. The book is well illustrated with screen captures, figures, tables, file listings and sample file segments.

The target audience leans toward NT administrators involved in UNIX integration. Nevertheless, UNIX administrators involved in NT integration will also find a wealth of information on providing SMB networking services to Windows clients.

There is a movement to publish a Common Internet File System (CIFS) file access protocol for Internet use. CIFS is based on the SMB protocol. Several sections of the book discuss using SMB and CIFS server resources lumping them together in several discussions. Visit the http://www.cifs.com/ web site for specific details and the status of CIFS.

SAMBA Black Book opens with a quick tour of networking, obtaining and installing Samba, and provides the steps for setting up Samba file and print servers. Next, Microsoft client use of Samba servers, configuring Win95/98/NT to use a SMB/CIFS server resource, and steps for using SMB/CIFS from a UNIX machine are presented. Following this are steps for using SMB/CIFS in complex network environments and using Samba as a domain controller.

Advanced topics include automating Samba, using it as a FAX server, and performance tuning. Solutions to complex mixed operating system networking challenges are offered as well. Examples of the solutions offered include browsing through multiple subnets, running databases on Samba servers, operations through an IP filtering firewall, user login authentication, Microsoft Windows roaming profile support, and backup and recovery of a mixed operating system network.

Next, steps for securing Samba servers and Samba troubleshooting are provided. Chapter 13 contains a collection of tidbits such as Samba bug fixes, some early words about Windows 2000 and Samba, and the forthcoming Samba Win NT printer support.

Information is well laid out and labeled. Major topic titles are written as statements of knowledge or a task to be performed. “Using NetBIOS”, “Understanding SMB” and “Setting Up a Share” are some example titles. Each chapter opens with the major topic titles repackaged as a list of Immediate Solution statements. “Setting Access Rights on Shares for Different User Groups” is one Immediate Solution statement from Chapter 4. The book presents only the information needed to understand the idea or accomplish the task. I feel this approach keeps the book very focused—the words lean and the content meaningful.

Readers knowledgeable about a subject area can easily skip over chapters or sections of chapters without hindering their implementation of Samba. I found myself reading the first few chapters sequentially, then began skipping around, reading only that which supported the task I was performing or information I was seeking. For example, I wanted to know what programs made up Samba. I found a detailed discussion of the Samba executables. Each Samba executable was first divided into one of three groups: servers, clients or utility. Next, the author presented the purpose of each program. Then he moved on to discuss the characteristics and use of the program's options. This explanation gave me a good feel for the location of Samba functionality.

Appendix A of this book lists sources for Samba and Samba documentation. Internal **smb.conf** constructs, parameters and legal parameter values are discussed. Appendix B is a delightful touch. It contains a list of additional resources, organized according to chapter and subject area.

Many acronyms are used in this book. The index did not help me when I wanted to find the first use of an acronym and its associated explanation. One example is CIFS. I remember reading about CIFS in several places, yet I could not find the acronym nor the phrase “Common Internet File System” listed in the index.

The accompanying CD-ROM reportedly contains a complete copy of the Samba FTP site and copies of configuration scripts contained in the book. Various programs mentioned in the book are also included on the CD-ROM. Binaries for some Linux distributions, commercial flavors of UNIX and related operating systems.

According to the book, binaries supporting 19 operating systems were included. I selected Novell, MVS, IRIX, AIX and Solaris operating systems from the list. I could not find a binary, or reference to a specific binary, for Novell, MVS or IRIX on the CD-ROM. Granted, this may not be a problem for most people, I mention it only to point out that there are some differences between what the book says is on the CD-ROM and what I could find on it. The latest release (at the time of pressing the CD-ROM) of the RFCs is included.

Technology often travels faster than book publishing. As the author suggests, visit the Samba web site (http://www.samba.org/) to obtain the latest release of Samba and its documentation. From what I read on the Samba 2.07 announcement page, the solutions offered in *SAMBA Black Book* should still apply.

Daniel Lazenby (d.lazenby@att.net) first encountered UNIX in 1983 and discovered Linux in 1994.

Advanced search

GNOME/GTK+ Programming Bible

Ben Crowder

Issue #79, November 2000

A solid introduction and reference to both GTK+ and GNOME programming.

- Author: Arthur Griffith
- Publisher: IDG Books Worldwide
- URL: http://www.idg.com/
- Price: $39.99 US
- ISBN: 0-7645-4640-6
- Reviewer: Ben Crowder

It all began with the GIMP. Years back, the developers of the famous image processing program realized that Motif (a toolkit for X Windows) wasn't as good as they had thought, but there wasn't much else out there to choose from. So they wrote their own widget set, GTK+. Then, later on, came GNOME, which wrapped around GTK+ and provided higher-level widgets (like the calendar and the calculator). It can all get rather confusing at times, which is where *The GNOME/GTK+ Programming Bible* comes in.

The book is a solid introduction and reference to both GTK+ and GNOME programming. It is liberally scattered with code (roughly a third of the book consists of examples and sample code, all of which is on the accompanying CD-ROM), and includes line-by-line explanations for most of the samples.

Chapter One introduces GTK+ and GNOME, explaining how GTK+ is object-oriented although it's written in C, as well as sketching out what signals and callbacks are. The second chapter is a tutorial on a very basic GTK+ program, a window with a button. It also covers how to create a basic GNOME window. In Chapter Three, you learn how GTK+ and GNOME pop-ups and dialogs work. The GTK+ method of packing widgets is, admittedly, different from most other windowing systems, and the fourth chapter covers just that by explaining packing boxes and tables. Virtually infinite flexibility comes with the ability to pack widgets inside of other widgets (i.e., tables within boxes within notebooks within scrolled windows within tables, ad infinitum). Chapter Five explains how this works, showing you the way to endless hours of packing pleasure. The glues that hold GTK+ together—events, signals and callbacks—are described in Chapter Six.One main point of GUIs is the ability to have graphical representations on the screen, <\#225> la icons and other pictures. Chapter Seven explains the XBM and XPM formats and how to use them in your programs. Finally, the eighth chapter covers menus and toolbars—stock menus, disabling/enabling menu items, and radio/toggle buttons in toolbars and menus.

Part II contains step-by-step instructions on a variety of GTK+/GNOME subjects. Chapter 9 covers the GnomeCanvas widget, Chapter Ten describes the GTKDrawingArea widget, and Chapter 11 explains how to use graphic contexts. Input devices are important in any type of interface, so Chapter 12 is devoted entirely to the mouse and the keyboard. Chapter 13 provides some basic instruction on using fonts, and Chapter 14 shares some tricks for working with widgets. GNOME applets are covered in the fifteenth chapter, and drag-and-drop is described in the sixteenth. MDI (Multiple Document Interface, used for having more than one document open at a time) is explained in Chapter 17. From time to time, you'll realize the widget you need doesn't exist. There may be something similar, but it doesn't quite do what you want. Chapter 18 shows you how to create your own widget, either from scratch or based on another one. Chapter 19 explores configuration (manipulating the config files found in your ~/.gnome directory) and internationalization.

The third part of the book is primarily a reference; and a very useful one at that. I would recommend this book for the third part and the appendices alone. For those who would rather compile it themselves instead of downloading a prepackaged version, chapter 20 explains how to install and build the GNOME source code. Chapter 21 is a list of GTK+ widgets, from **GtkAccelLabel to GtkWindow**, with information on each (such as inheritance, function lists and example code). Chapter 22 does the same for GNOME widgets. While GNOME is multi-platform, it is primarily used under Linux; thus, there are some Linux-specific issues which are covered in Chapter 23. Finally, for those programmers migrating over from Windows, Chapter 24 provides a comparative, point-by-point study of how Win32 and GNOME programming differ.

The appendices are quite useful as well, and they are where much of the meat in the reference section is located. The first appendix simply describes the contents of the CD-ROM, while the second explains how to set up your machine for software development with GNOME. The third appendix lays out the inheritance for each widget in GTK+ and GNOME, the fifth explains how to set and get arguments, and the fourth covers enumerated types used in GTK+ and GNOME (such as GTK_ACCEL_SIGNAL_VISIBLE). The sixth appendix lists all the signals, and the final appendix provides a useful list of functions, sorted by return type.

Is it worth the money? If you're interested in programming with GTK+ and/or GNOME, then the answer is a resounding yes. And even if you're not, it's a cool-looking Linux book that can sit on your desk and make you look smarter.

Ben Crowder has been heavily involved with computers for the past ten years, in almost every aspect (programming, graphics, networking, music and just about anything else you can think of). He has been working with Linux for the past two and a half years and has loved every second of it. In his spare time, he enjoys reading, writing, music and tweaking things on his Linux box. He currently lives in Utah and can be reached at crowder@netbrick.com.

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #79, November 2000

Readers sound off.

Thanks for the Samba

Your article on Samba in the August 2000 issue of *Linux Journal* motivated me to get my Samba tuned up and running right.

I also use NT 4.0 so there were some differences, and I ended up using a web article: www-4.ibm.com/software/developer/library/samba, by Daniel Robbins.

I realize you were describing your experiences with Red Hat 5.2. Your article did not describe what version of Samba you were discussing. I upgraded to 2.0.7 (RH wtmp).

One of the areas with which I had trouble was that Samba seems to take the first, or perhaps a random stab, at the interface on which it will run. The first try used my external DSL interface—not what I wanted. I found, in the Robbins article discussed above, that you could specify a global of interface = eth1 and that solved problem. Your article did not hint at the possible problem.

Robbins also recommended the use of “guest” rather than “nobody”, and that too solved a problem of the explorer not being able to see shares, even though net use/view could. PS: not many of the references discuss Internet use as did your article.

Thanks for getting me motivated.

—Paul Campbell seapwc@halcyon.com

Artists' Guide to Linux Desktop

I enjoyed Michael Hammel's *Forum* series, “The Artists' Guide to the Linux Desktop”. Having explored window managers in much the same way as Hammel, I was pleased to discover that someone else on planet Earth had reached similar conclusions about their various merits and defaults.

I would, though, like to point out an inaccuracy in Hammel's discussion of FVWM2. I imagine he'll thank me for it. FVWM2 does not, as he states, require the user to restart the window manager whenever a menu is changed. FVWM's “Read” command will reread and install the altered version of a menu file without requiring a window manager restart. The “Read” command, like all FVWM commands, can be bound to a menu, pop-up, key press, or mouse click. It can also be embedded in a function. The only thing to beware of is that the menu file should begin by destroying itself and any submenus (“DestroyMenu <name>”). Otherwise, the “Read” command will concatenate the new version with the old.

—Peter Schaffter df191@ncf.ca

Ads and Content

This is a reply to the letter from Tirath in the August issue. I think he is off base to complain about advertising in *LJ*. The rise of advertising has not harmed the editorial content as far as I can tell. In fact, it seems to me there is more editorial content than ever.

Far from being “junk”, as he call them, the ads let us know what companies, products and services are available to get things done in a Linux environment. In addition to the high information value of these ads, they also highlight the growing support for the Linux community and for *LJ* in particular. These companies are helping *LJ* to grow by the money they spend on advertising. Furthermore, by releasing products and services, they are helping Linux to grow. In my opinion, *LJ* is a perfect venue to make known such products and services. We ought to do business with these companies whenever possible.

—Bryan S. Tyson bryantyson@earthlink.net

Overseas Subscription Price Increase

As an international subscriber to *LJ*, I was shocked to see the international subscription price in the August issue. It appears to have risen from $37 to $62 per year. I understand that distribution costs are involved, however this seems a large increase, considering the cover price remains the same. I would appreciate your comments.

—Brian Galbraith brian.galbraith@bigfoot.com

Ever increasing postage rates and increased paper and production costs have forced us to raise our international subscription prices in order to keep offering them. Over the past few years, Linux Journal has lost a substantial amount of money on these subscriptions and, therefore, had to make a decision to increase rates to bring us close to a break-even point. We sincerely wish we did not have to pass this cost along to our international subscribers, but it was the only way we could continue to offer these subscriptions.

—Editor

(Not So) WordPerfect Office 2000 Review

I have some disagreements with Jon Valesh's review of WordPerfect Office 2000. Nearly the first page and a half was devoted to the politics of office suite development for Linux. I'm not a big Microsoft fan either, but I buy software based on whether it effectively solves an existing need. I'm frustrated by having to reboot to Windows to deal with spreadsheets and documents that I have to use for my job...the question is whether Corel can help me solve this problem.

Mr. Valesh had very few installation problems. My experience was the opposite. Installation under SuSE 6.3 was a nightmare. I'll omit the details in the interest of time, but even after installation, the software regularly crashed or froze. I noticed that Corel's latest ad in your magazine mentions compatibility with “major” Linux distributions, but no longer specifically mentions SuSE. (On the software packaging, SuSE is mentioned as a compatible distribution.) This actually reminds me of my past experiences with Microsoft—get the product out now, whether it works or not, and then fix it later. And no, I certainly don't expect reviewers to try software on each Linux distribution, but it might not be a bad idea for the software developers.

I've switched to Linux-Mandrake 7.1, and I tried WP Office 2000 again with better results. However, there are still some major annoyances. For example, when I hit “File/Open”, the dialog box opens *behind* the document window. Mr. Valesh calls these problems “idiosyncrasies”. I call them annoying bugs. I wish the review had focused more on how the software works, and less on installation and philosophy. Then, people could make an informed decision on whether to purchase the product.

Meanwhile, I'm hoping that Corel offers some kind of an upgrade deal to registered users when they finally get the “idiosyncrasies” fixed. —Jim Mueller jmueller@advancenet.net

Font for Letters Column

I would just like to point out that using a sans-serif font for the Letters section of *LJ* is not the best choice; more and more letters are including URLs, file names, or other pieces of verbatim text where it is important to distinguish between capital “I” and lowercase “L”. The August Letters, for instance, contains two references to the iclint rpogram, one as simply “Iclint” and one as ”http://Iclint.cs.virginia.edu”.

If you're reading this message in a mail reader that happens to use a sans-serif font, then you've probably gotten the point already. If not, let me recast those as “#c#int” and “http://#c#int.cs.virginia.edu”, and pose you the hypothetical question of how you would feel about that letter if you'd never heard of iclint before, and had no idea whether you were reading about something called “lclint”, “lclint”, “Iclint”, or “Iclint”. If it were me, knowing that most stuff in UNIX is case-sensitive, and knowing that host names are case-insensitive, I'd probably think you meant “lclint”, in which case I would be pretty frustrated if I tried to access that web site.

The same logic, of course, applies throughout your magazine, but I spotted it in the “Letters” column, so that's what I'm griping about.

—Cloister Bell cloister@hhhh.org

June 2000 Issue

I was excited when I felt how thick the June issue of *LJ* was, but when I opened it, I was a bit disappointed, ho hum how boring—well not all of it. But I long for the days when *LJ* used to feature exciting events and accomplishments like Robots, Linear Accelerators, etc.

I am sure you know *Byte* and *Circuit Cellar Ink* and its history; how about starting a section related to electronics and interfacing with Linux, various embedded hardware, etc.?

—Ross Linder ross@mecalc.co.za

Ross, you are in luck. We have been watching the embedded use of Linux grow and have decided to seriously address it. The September 2000 LJ focused on the embedded use of Linux. We also have a special supplement that will be mailed to our subscribers in October with a new magazine to follow in 2001.

—Editor

Quicken for Linux and Alternatives

I was very interested to read your Focus Editorial and the “Linux Finance Programs Review” (August 2000). I was relieved to find I am not the only one who is addicted to Quicken. There seems to be plenty of support on your side of the Atlantic.

I am currently trying to make a transformation from using “The Other Operating System” to Linux. I have two problems. One that I can do nothing about is my employer's use of NT, and a need to work at home. The other is that I do not seem to be able to manage without Quicken, and there does not appear to be a viable alternative for Linux.

Ralph Krause has done an excellent review of the possibilities, but none of them sounds like a viable full alternative. Each has some of the functions of Quicken, but not all. None seem to deal with an investment portfolio at all.

As you presumably know, Quicken is written here in England by Intuit Ltd. I have just written to them to ask when (not if!) they intend to port Quicken to Linux, drawing their attention to your articles. I will let you know when (or if) I get any response.

—David R. Hignett Hignett@dial.pipex.co.uk

September 2000 Issue

I have used Linux for about five years now, starting with a simple umsdos installation. I have no computer training, and so at times it has been a struggle. The people involved in the “movement” have fascinated me as much as the system, and have made it fun to participate even in my little way. Your recent forays into presenting those personalities have been fun for me, and I particularly enjoyed this first installment with Phil Hughes. Keep it coming.

—Gary Dolan fred1@inebraska.com

Return of the Python!

I was shocked and dismayed, not by the cover, but by the response from some of your readers to the picture on the front of the Python Supplement (May 2000). Such a picture would barely raise an eyebrow here in Australia!

I find it to be a never-ending source of wonder that a nation that consumes so much of the world's resources and gives so little back in return can occupy itself so completely with this kind of moralistic debate. If the Puritan heart of middle America still can't reconcile themselves to the fact that the price of freedom is tolerance, then I strongly suggest they hand in their semi-automatic weapons and 454 cubic-inch pick-ups!

We have nothing to fear from depictions of the human body, only from the mindsets of those who want to use such images to peddle their simpilistic, narrow-minded views of the world.

—Arthur Watts arthur.watts@gbst.com

I know we promised we had published the last of the responses to the naked cover, but we couldn't resist.

Disgusted

In reference to the letter titled "Disgusted" in your September issue:

While I'm not as outraged as Dale Lakes, I must say I'm deeply disappointed to find out *Linux Journal* is using NT and IIS for anything. I realize you may be having problems coming up with a Linux solution, but just think of the field day Microsoft is going to have when they find out that one of the largest "Voices" of the "Linux Community" runs their e-commerce site off of NT and IIS, because they can't find a suitable Linux alternative.

—Howard Pepper hpepper@mindspring.com

UpFRONT

Various

Issue #79, November 2000

Stop the Presses, *LJ* Index and more.

FREELY CORRECTED

In my interview with Craig Burton in the August 2000 issue of *Linux Journal* ("Uncollapsing Open Source Distinctions: Talking with Craig Burton", p. 16), I said "historically, the Open Source movement has tried to move away from the Free Software movement's anti-commercial rhetoric and policies."

A few days ago I got an e-mail from Richard Stallman, under the subject "Who's anti-commercial?". His answer: not him. And not the Free Software movement, either:

> We do not have anti-commercial rhetoric or policies, and I'm surprised you would say this.
>
> I know why some people say we are "anti-commercial". We criticize a common business practice, and people who do that are often accused of being "anti-commercial". But, the fact is that if a program does not allow commercial use, or if you can't sell copies, we reject it.
>
> We do not compromise our principles to cater to business; business today is so used to such treatment that anyone who stands firm when business says "Change!" is likely to be called anti-commercial. For instance, the GPL is designed to prohibit some anti-social practices, and this applies to business just as to individuals and schools. If people say the GPL is not "business-friendly", they probably mean it dares to say no to some of the things their businesses want to do.
>
> But the GPL extends the same rights to business as it does to everyone else. And we try to cooperate with business in ways that are consistent with our

> principles. For example, I asked publishing company people for advice when writing the GNU FDL.
>
> So would you please post a correction to that statement about us?

Since I was making distinctions between two movements, I decided to share Richard's correspondence with the prime mover of the other one, Eric S. Raymond, hoping to triangulate a bit on the full extent of my error. “Okay,” I wrote, “did I step in it here, or (so far as what he quotes) am I right? Or sort of right?”

ESR wrote back, “It's a tough call. No, the official interpretation of FSF doctrine is not anti-commercial. In that sense, yes, you stepped in it.”

An on-the-other-hand explanation (OTOH) followed, but it's not one that RMS found agreeable. Nor was RMS's disagreement with ESR's OTOH agreeable with ESR. The e-mail volley between the two gentlemen continues to fill my in-box, so I'll leave that one alone for now.

Meanwhile, I invite readers to visit the Free Software Foundation site at http://www.fsf.org/. Here is part of the FSF's explanation of free software:

> “Free software” is a matter of liberty, not price. To understand the concept, you should think of “free speech”, not “free beer.”
>
> “Free software” refers to the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom for the users of the software:

- The freedom to run the program for any purpose (freedom 0).
- The freedom to study how the program works and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. (freedom 3). Access to the source code is a precondition for this.

A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, with or without modifications, gratis or charging a fee, to anyone anywhere. Being free to do this means (among other things) that you do not have to ask or pay for permission.

So the statement, and I, stand corrected. For my thoughts about related matters, see this month's “Linux for Suits” on page 20.

—Doc Searls

TRENDS

by Reginald Charney

Languages

Over the last 15 months, the languages with the most growth have been those directly related to the Internet. Thus, XML, Perl, HTML, and Java have flourished. However, even these high-flyers have suffered reversals in the last few months (see www.accu-usa.org for more details). It is interesting to note that the highest flyers are also showing the most deceleration in demand.

Platforms

One of the major changes in the demand deceleration of platforms is that Windows 2000 has now joined the other platforms that are experiencing decreased demand. Interestingly enough, while demand for individual flavors of Windows has decreased, overall demand for Windows is still growing, albeit slowly. This includes all dialects of Windows.

What's Hot, What's Not

You must have been on an island not to know that LinuxWorld was in San Jose in August. It had so many exhibitors that some of them overflowed the exhibit hall and ended up out in the hallways. Rumor has it that next year's West Coast LinuxWorld will move to San Francisco's Moscone Center to handle the anticipated larger crowds. There also was a lot of big names there: IBM, Dell, Compaq SGI, HP, Sun, as well as the usual Linux crowd, such as VA Linux and all the major Linux distributors. The BSD folks also had an interesting presence.

If anything can be said about future trends, it has to be Linux for embedded systems. In contrast, last fall's theme was Linux in the business world, and this year delivered great growth in that arena. At this show, most of the software exhibitors were providers of B2B solutions. Two products caught my eye: the color PDA, Compaq iPaq, running PocketLinux; and the black and white Helio. At $500 and $200, respectively, they will give Palm a real run for its money (http://www.PocketLinux.com/). eGrail (http://www.eGrail.org/) is an open-source provider of content management software for the Web. It provides server-based access to the central repository and tools through any browser.

THEY SAID IT

“Five years from now the government will measure open-source project starts the way it now measures housing starts.”

—Bill Weinberg, MontaVista

“In any business model you need someone to sue. That's the American way.”

—Bill Weinberg, MontaVista

“Advice is what we ask for when we already know the answer but wish we didn't.”

—Erica Jong

“The condition of the free man is that he does not live for the benefit of another.”

—Aristotle

“They rely on customers to find uses for minicomputers, rather than burdening the company with huge costs of developing and marketing applications of its own. Digital salesmen, engineers selling to other engineers, nurture strong and lasting relationships with customers...it's surprising how little they've caused their own growth. For years, they've been dragged along by interesting applications their customers came up with.”

—From *In Search of Excellence*, by Tom Peters and Bob Waterman (1982), on the subject of Digital Equipment Corporation.

“Time is an illusion. Lunchtime doubly so.”

—Douglas Adams

Disease can be cured; fate is incurable.

—Chinese proverb

“Prediction is especially difficult. Especially about the future.”

—Niels Bohr

“It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change.”

—John McFee

“If by some fiat I had to restrict all this writing to one sentence, this is the one I would choose: The summit of Mount Everest is marine limestone.”

—John McFee (on geology)

“I am the last bastard of free speech.”

—Howard Stern

Takeoffs are optional. Landings are mandatory.

—Sign in small airport

“The average knowledge worker will outlive the average employing organization. This is the first time in history that this has happened.”

—Peter Drucker

“Imagine if every Thursday your shoes exploded if you tied them the usual way. This happens to us all the time with computers, and nobody thinks of complaining.”

—Jeff Raskin

“It's a damn poor mind that can only think of one way to spell a word.”

—Andrew Jackson

“What is wanted is not the will to believe, but the will to find out, which is the exact opposite.”

—Bertrand Russell

“All models are wrong. Some models are useful.”

—George Box.

BUZZPHRASER GOES OPEN SOURCE

The original BuzzPhraser was created by the staff in the office of my old company in 1990 or so. It took the form of a spreadsheet that we gradually filled with overused buzzwords. To qualify, they had to work in a phrase the way a blank tile works in Scrabble: you can use it anywhere, but it has no value.

We sorted these words into a series of columns: adverbs, adjectives, adnouns (nouns used as adjectives), nouns, prefixes and suffixes. After the columns began to overflow, I thought “Hmm...BS seems to be programmable. Let's make something here.” So I got together with Ray Miller of Turtlelips Productions and we (mostly Ray) crafted a Hypercard stack that put together random buzzphrases from the table, based on various user-defined combinations of modifiers, nouns, prefixes and suffixes. BuzzPhraser made a bit of news and then quickly became one of the most-downloaded files on both AOL and Compuserve.

Several years ago, Charles Roth (the prime author of *Caucus*, the primo conferencing software) kindly adapted BuzzPhraser to the Web, where it has served ever since, burping up useful but value-free generica for publicists and their enemies everywhere. Examples:

- management technology implementation channel protocol
- empowerment architecture topology
- workgroup-dependent program-level i-shrinkage rule operation
- structured client leadership chain
- substantially phase-free product-intensive demand-elegant gesture exchange
- enhanced policy services content dependency solution leverage policy

Now Charles has completely rewritten the entire site in Javascript and open sourced it. To get the source, go to http://www.buzzphraser.com/. And let us know what you do with it.

—Doc Searls

LIKE WE SAID, ONLY CHEAPER

In the August 1999 issue of *Linux Journal*, Bruce Fryer wrote a piece called “Thinkful Wishing” that envisioned “a Linux-based iMac”. Here was his design: “Use nothing but end-of-life components with no fan and no floppy so it can be built dirt cheap. Boot from the CD-ROM drive. Solder everything onto the boards including memory, with the possible exception of a communications slot. We're talking about a black and white box here: simple and reliable as an old phone. Shoot for a $500+ price point, complete with all network connections, OS and software, including management agents. Put the money in memory and display, which ought to be active matrix, if there's any way.” It goes on, but you get the drift.

Pretty soon you will be able to get one, pretty much to that spec, from New Internet Computer, or NIC, http://www.thinknic.com/. If that name sounds vaguely familiar, perhaps its because yes, indeed, this is the latest incarnation of Oracle CEO Larry Ellison's thin client. While the original “NC”, or Network Computer, was conceived as a kind of terminal, the NIC is a Linux workstation. It boots Linux 2.2 off the CD-ROM, along with Netscape Navigator 4.7 and a catalog of plug-ins. The hardware is a roster of generics: 266MHz processor, 64MB RAM, 24x CD-ROM, 2 USB ports, 10/100 base T MB Ethernet, keyboard, mouse, speakers. Applications and storage will reportedly be handled by a remote server (though it can save to “a capable third-party Internet storage system”. It will run Windows applications too, through the Citrix MetaFrame client. Price: $199.99 (US). Bundled with a 15'' (800 X 600) monitor, $329.98. The company offers a free ISP, or you can use your own.

The CEO of The New Internet Computer Company is Gina Smith, the high-profile veteran technology columnist, radio and TV personality. Smith's most recent gig was cohosting CNET News.com. The scope of the company's ambitions is apparent in two facts that appear to guarantee a market for the boxes. One is Larry Ellison's stated commitment to spend $100 million to computerize schools. The other is Oracle's recent donation of 500 NICs to the Chicago Public School System, with a promise to match another 500 donated by the National Association of College and University Business Officers (NACUBO), which is chaired by General Colin L. Powell, USA (Ret). In May, Oracle also donated 1200 NICs to the Dallas Independent School District.

—Doc Searls

BLOCKING DOUBLECLICK.NET

Theory: If you tell your name server it is in charge of the domain “doubleclick.net” then it will happily answer all requests for “Where's doubleclick.net” with the smug reply, “I know everything there is to know about doubleclick.net, and I can tell you with complete confidence that there is no such place.” If browsers can't find doubleclick.net, then doubleclick.net can't track those users.

Because many users typically use each name server, this is not only one of the fastest ad blocking techniques known to freedom-loving humanity, it's also the technique that protects the most users.

1. Log in to the name server as root.
2. Find your named.conf file. It may be in the /etc or /etc/bind directory. If you have trouble finding it, use this command:

```
find / -name named.conf
```

1. Open the named.conf file for editing in your favorite text editor. Locate the “localhost” zone. It should look something like this:

```
zone "localhost" {<\n>
 type master;
 file "/etc/bind/db.local";
};
```

It doesn't matter if the file name on the line beginning with “file” is different.

Make a copy of the localhost zone elsewhere in the file. Change the copy to read “doubleclick.net” instead of “localhost”.

```
zone "doubleclick.net" {<\n>
 type master;
 file "/etc/bind/db.local";
};
```

Save the file and exit the text editor. If you mess up the file, exit without saving and do step 3 again.

Step 4. Find out the process id of named with this command:

```
ps ax | grep named
```

Let's say you get something like this:

```
7907 ?        S      0:03 /usr/sbin/named
```

Do the following:

```
kill -HUP 7907
```

Use whatever process ID your named has, not “7907”. You're done. Clear your browser cache and rejoice!

—Don Marti

LJ INDEX—NOVEMBER 2000

1. Number of Britain's FTSE 100 companies that have no web site or cannot be contacted by e-mail from their web sites: **29**
2. Number of U.S. Fortune 100 companies that have no web site or cannot be contacted by e-mail from their web sites: **23**
3. Percentage of the remaining 71 FTSE 100 companies that fail to respond to multiple requests for basic investor information even after three months: **20**

4. Percentage of the remaining 76 Fortune 100 companies that fail to respond to multiple requests for basic investor information even after three months: **33**
5. Number of ways "ough" can be pronounced in English: **9**
6. Year when the first traffic accident occurred: **1896**
7. Number of traffic accidents at the intersection of Wilshire and Santa Monica Boulevards in Beverly Hills in 1998: **242**
8. Rank of that intersection among most dangerous in the U.S.: **4**
9. Position of less than $25K per year (U.S.) households among all income groups in Internet usage: **1**
10. Annual rate of increase for less than $25K per year U.S. households: **50**
11. Position of less than $25K per year households among all income groups in time spent online: **1**
12. Age at which William Roseman became the youngest elected official in New Jersey: **18**
13. Age of William Roseman at the time he founded Linux Global Partners: **40**
14. Amount invested in Linux start-ups by Linux Global Partners as of September 2000: **$25 million**
15. Number of Linux start-ups in which Linux Global Partners has invested: **8**
16. Amount invested by Linux Global Partners in Helix Code: **$2,200,000**
17. Helix Code quarterly earnings expected by William Roseman by 2002: **$30,000,000**
18. Typical number of new Helix Code downloads and installations, per day: **4,000**
19. Combined ages of Helix Code's cofounders: **49**

Sources:

- 1-4: The Rainier Web-Index Study
- 5: viz: "A rough-coated, dough-faced, thoughtful ploughman strode through the streets of Scarborough; after falling into a slough, he coughed and hiccoughed."
- 6-7: *USA Today*
- 8-11: Media Metrix, Inc.
- 12-13 and 15: *San Jose Mercury News*
- 14 and 18: National Public Radio
- 16-17: *Boston Globe*
- 19: *San Jose Mercury News and Business 2.0.*

THIS MONTH IN LINUX JOURNAL SIX YEARS AGO

It was six years ago this month that Phil Hughes wrote an article titled "Selecting Hardware for a Linux System". Dust off the copy from your archives, or read it at the *LJ* web site (http://www.linuxjournal.com/lj-issues/issue7/2850.html).

STOP THE PRESSES: HARD REAL TIME

While Linux has quickly emerged as a winning embedded operating system, the main reason has never been speed. In fact, some forms of embedded Linux, such as Lineo's popular Embedix, make Linux a "hard" RTOS (Real Time Operating System) by combining it with a second kernel built for real-time work. MontaVista Software, Inc., however, has been implementing Linux itself as an RTOS since the company was formed early last year, and has worked to improve the performance of the Linux kernel to the point where it qualifies—all by itself—as a "hard" real-time OS. At press time (September 9) for this issue, MontaVista has delivered what it claims is a hard real-time Linux kernel, based on the current 2.4 code version. The company claims that this new kernel is fully preemptable, and provides a 30-fold improvement in performance over the 2.4 base. Claimed worst case application responsiveness will be in the hundreds of microseconds. In an interview for *Embedded Linux Journal* (this month's supplement to *LJ*), James Ready, Montavista's president & CEO, said, "This isn't just a better embedded Linux. It's a better Linux."

As for the new kernel itself, Ready explains that the real issue is not just responsiveness to interrupts, but preemptability. Also that making Linux preemptable involved leveraging the same engineering work that yields Symmetric Multi-Processing (SMP).

Hard Hat Linux is MontaVista's embedded Linux brand. The new version, which is expected to come out of peer review by January, will offer a real-time scheduler that features deterministic real-time application selection and dispatching, without altering standard Linux scheduling. There will also be an optional interrupt accelerator, based on technology from RTLinux, which is a product of FSMLabs of Albuquerque, New Mexico. This has a 5x claimed interrupt responsiveness improvement. The first prototype is for IA32 platforms and was available immediately at press time from ftp://ftp.mvista.com/ A technical paper explaining MontaVista's real-time developments is available at ftp.mvista.com/pub/Real-Time/2.4.0-test6/preempt.txt. Read the interviews with both Jim Ready and Lineo's Bryan Sparks in the current *Embedded Linux Journal* also see responses to MontaVista's announcement by TimeSys and Lineo on page 22.

—Doc Searls

Jason's Tips

Jason Schumaker

Issue #79, November 2000

Tips and factoids from Jason Schumaker.

Link of the Month:

Learn the hardware and software procedures for making MP3 and Linux get along. www.ssc.com/mirrors/LDP/HOWTO/MP3-HOWTO.html

Rebuilding Your Kernel

When rebuilding kernels, make a backup of the last one that worked, and include a stanza in your /etc/lilo.conf to allow booting the working kernel. For example, with kernels found in /boot/vmlinuz and last known working kernel in /boot/vmlinuz.works, /etc/lilo.conf might look like:

```
boot=/dev/hda<\n>
install=/boot/boot.b
map=/boot/map
vga=normal
timeout=200
prompt
read-only
image=/boot/vmlinuz
root=/dev/hda1
label=Linux
read-only
password=Vogons
restricted
image=/boot/vmlinuz.works
root=/dev/hda1
label=Itworks
read-only
password=Vogons
restricted
```

this will present a LILO prompt and wait 20 seconds for instructions as to which kernel to boot. The user may type

```
Linux
```

or

```
Itworks
```

to select most recent build or most recent known good kernel. The “password” and “restricted” parameters prevent entry of additional boot parameters which might compromise security, without entry of the correct password. Make /etc/lilo.conf read/write to root, no permission to group or other.

Once a kernel is known to work, as root:

```
cp /boot/vmlinuz /boot/vmlinuz.works<\n>
lilo
```

Dan Wilder

Hardware Tips

If you are using Linux and your system keeps crashing suspect the hardware, unless you are running some bleeding edge development kernel.

First, check the memory. There is a great program for memory check called **memtest86** (http://reality.sgi.com/cbrady_denver/memtest86/).

The memtest program is an x86 boot sector. You can put it on a floppy and boot the suspect computer to memtest from a floppy drive. Or, my favorite is to load memtest from lilo as an operating system. Use an image clause in your lilo.conf file like this:

```
image=/vmlinuz          #Your Linux kernel<\n>
        label=Linux
        read-only
image=/memtest          #Your copy of memtest
        label=Memtest
```

Have You Checked Your CPU Lately?

Sometimes CPU's just don't perform well at their rated speeds. You run into weird things like segmentation faults while using **ls**! Under clock your CPU. A good way to give your CPU a test is to run **setiathome** on your machine. See: http://setiathome.ssl.berkeley.edu/. This puts your CPU to work doing lots of fast Fourier transforms. Don't forget to join the *Linux Journal* setiathome group.

Still having problems? Lower the motherboard bus speed by a hair. If you see or smell smoke, shut off the power.

—Rory Krause

VI TIP: Converting DOS files to Linux/UNIX

It happens often: an author sends a file created in DOS, which adds ^M throughout the text. To globally remove all ^Ms from a DOS file I use one of the following two options:

```
dos2unix <filename>
bni:or
```

```
:%s/^M//
```

The ^M is produced by typing **CTRL-V**, then **CTRL-M**. Use them both—for a bit of variety!

—jason schumaker

Not a Rhetorical Question

“Linux happened without the help of deep pockets—how can we keep the magic?”

—Bruce Pehrens, posing a question to Michael Dell at LinuxWorld.

“Linux has to change the world more than the world needs to change Linux. Because IT sucks.”

—Don Marti

Looking for a news site similar to Slashdot? Try: http://www.kuro5hin.org/. Reader's vote articles on or off!

HIGH-TECH NEWSFLASH

Palm will soon be unveiling a new version of the Palm Vx—the Claudia Schiffer model! Try http://www.claudiaschiffer.com/ for more details.

QUOTE

“If Jesus Christ would have lived long enough, he would have repeated himself, too.”

—Kurt Vonnegut Jr.

Anniversary for Red Hat IP0

How time flies! It has already been a year since Red Hat went public. If you had bought yourself a crispy new share on August 11, 1999 for $26 at the trading day close you would have taken a roller coaster ride up to $151 and right back down to about $25 today.

Publisher's Announcement

Phil Hughes

Issue #79, November 2000

I'd like to introduce a *Linux Journal* supplemental issue which will hit the streets October 10, 2000: *Embedded Linux Journal*.

In 1968, when I got my first job in computing, we didn't call that room full of computers with a disk drive the size of a Volkswagen an embedded system. But it was. I worked for Collins Radio and what we were working on was a message-switching system. Today, however, that same computing power could easily fit in a 1U rackmount box or be implemented on a Netwinder or Cobalt Qube.

The point is that calling a system embedded doesn't have anything to do with its size, but whether it performs some dedicated task. Besides the changes in size over the years, there have been cost changes. While my microwave doesn't have an embedded processor in it, most, do as do most traffic light controllers and virtually every printer in the world.

Doing an inventory of what is around me at home, here is my list of things that I know have embedded processors: Palm organizer, cell phone, FAX/answering machine, scanner, digital camera, video camera, dish TV receiver, VCR, stereo, laser printer, DSL modem car. At work I can add microwave, label maker, phone system, voice-mail system and conference phone.

This doesn't count other items that most likely have them as well: disk drives, tape drives, monitors, TV and clock radio. This is a big change from 1968. With $50 (US) products out there in the embedded market, there is a lot more to consider than just making a product that works.

We want to help you take the next step. Hardware costs have fallen dramatically, making it possible to put computers into relatively inexpensive products. Efficient code can reduce RAM and ROM requirements. But there are additional costs besides hardware. The OS for your product, development time, development tools and licensing all cost money. Shipping a product with bugs can cost you money and reputation.

With that I'd like to introduce a *Linux Journal* supplemental issue which will hit the streets October 10, 2000: *Embedded Linux Journal*. In this special issue you can look forward to conversations about:

- industry news—emphasizing open-source software solutions
- reviews of products to reduce development time and improve testing
- case studies that will save you time
- design solutions that show you why embedded Linux is the cost-effective answer
- hardware vs. software considerations

Current *Linux Journal* subscribers who live within North America will receive this special supplement at no additional charge. This issue will also be heavily distributed at upcoming trade shows, other industry events, and to targeted mailing lists.

We're certain you'll enjoy this upcoming *Embedded Linux Journal* supplement. We look forward to your feedback!

Sincerely,

Phil Hughes Publisher

Best of Technical Support

Various

Issue #79, November 2000

Our experts answer your technical questions.

Something Odd

When I installed my Mandrake 7.1 system, I partitioned it so that /dev/hdb4 would be mounted to /usr and /dev/hdb2/ on / and /dev/hda1 on /boot. My problem is when I restart or halt Linux at the stage when it unmounts the file systems, I get this error: /usr: device is busy. And then it freezes there. When I start the system again, it takes a long time to check it dev/hdb4. I would like to know how to fix this. —Charles Diaz-Alejandro, oguara@bellsouth.net

It sounds like there must be a process that is still running when the system drops runlevels. If that process has any files open or accesses to that partition, it is possible that **mount** will not be successful. What you might try booting into single-user mode and then looking at how many processes you have and which ones they are specifically. Runlevel 1 should have very few processes because they get killed when you switch runlevels—it is possible that one of the processes is not responding or cannot be terminated properly. What you could try doing while in single-user mode is to remount the /usr partition read-only with **mount -o remount,ro /usr**. If this fails due to “busy” errors, then try killing off some of the higher-numbered processes. You probably don't want to kill the very low-numbered ones, though. After each kill, try to remount the /usr read-only until it succeeds. Then you will know which process is causing the hangup. It may also be due to the amd process. I have seen amd and nfs cause mishaps like this. Once you know which process it is, you can disable the service or at least work towards getting it fixed properly. —Andy Bradford, andyb@calderasystems.com

The system prefers not to unmount a file system if a file within it is opened or if a process has its current working directory within the file system. Most distributions try to remount the file system read-only if umount fails, in order to avoid the file system check pass on next boot. Mounting read-only fails only if a file in the partition is opened for writing. If your mount command is not to ancient, change the **umount** to **umount -r** (manual says: “In case unmounting fails, try to remount read-only.”) I can't tell why your umount fails, but you can try to find the guilty process by calling “fuser” in the shutdown script before the umount command:

```
fuser -m /usr; ps afx; sleep 10
```

(The “ps” helps in understanding what the process IDs are). Refer to man fuser for any details. —Alessandro Rubini, rubini@linux.it

It's probably due to a minor bug in Mandrake. One way to find out is do add this just before the end of /etc/rc.d/init.d/halt:

```
(fuser -vm /usr; ps auxww )| more
read a
```

(before the “# Now halt or reboot” section). This would usually require you to copy /usr/sbin/fuser to /sbin/fuser to that it's accessible, but since /usr is not being unmounted, it should still be accessible in /usr. Fuser should show you a list of processes that still use the /usr mountpoint and you can look up their number. Then, depending on the output, you can modify the halt script to kill those processes (although the halt script should already kill everything) or ask MandrakeSoft why those processes don't get killed and your partition doesn't get unmounted. —Marc Merlin, marc_bts@valinux.com

“Sleep” Mode

Does Linux have a “standby” mode similar to Windows 98? This would enable my computer to sort of power down and sleep when no activity is detected. —Ronnie Bell, ronbell@cais.com

If your kernel has APM compiled in (and I believe the default Red Hat 6.2 kernel does), you can use the standby button on your PC case if you have one and configure it to put the machine on standby if you do a short press. From the command line, you can also use **apm -s** or **apm -S** for suspend and standby. —Marc Merlin, marc_bts@valinux.com

The kernel automatically halts the CPU when there is no work to be done, enabling the CPU to enter a low-power mode (the CPU is reactivated when there's more work to do; this is all transparent to you). You can also configure APM (Advanced Power Management) support in your kernel, assuming you have a laptop with an APM-compatible BIOS. For even more power savings, you can tell the X server to use DPMS (the Display Power Management System) to turn off your monitor after a long enough period of keyboard/mouse inactivity. I do that with:

```
xset +dpms
xset dpms 600 1800 3600
```

This puts the monitor in standby mode after 10 minutes (600 seconds), goes to suspend mode after 30 minutes (1,800 seconds) and turns it completely off after 60 minutes (3,600 seconds). Each of these levels increases the power savings. Adjust the numbers to suit your working style, but be aware that your video card and monitor must support DPMS in order for this to work—otherwise, nothing special will happen. As an alternative to the xset command, you can enable DPMS support in your XF86Config file; search for “power_saver” in the XF86Config man page for more information. Finally, you can use the **hdparm** command to spin down your hard disk. I don't personally recommend this, if only because spinning the drive down and back up reportedly causes as much wear and tear as six hours of use, but you can do it if you want to (read **man hdparm** for details). I'm willing to make some sacrifices for the good of the planet, but sacrificing my Linux disk is simply too much to ask. :-)—Scott Maxwell, maxwell@ScottMaxwell.orgi

You should use the apmd package, which is available from Red Hat's distribution. Of course, the “apmd” d<\#230>mon should be started at boot time (try “setup” or “linuxconf” to add it to the system services). —Pierre Ficheux, pficheux@com1.fr

Who Am I?

I'm running Red Hat 6.1 on a Dell machine and I have DSL Internet access through Telocity. Until I got the Telocity service, my computer always thought of itself solely as “localhost.localdomain”. (For example, my bash prompt would read “[jenny@localhost tmp]$ ”). This seemed lame, but caused no problems. After Telocity came into my life, it mostly thought of itself as dsl-216-227-xxx-xxx.telocity.com. I'm not completely sure, but I think my prompt changed accordingly. After a recent reboot, though, we're back to “localhost.localdomain” and certain pieces of software with license managers that use the server's name are not working anymore. The licenses had been set up during the DSL phase. I've gotten them working again, but I'm afraid that one day my machine will go back to being dsl-blah-blah-blah and I'll have to fix it all again. Where does my computer's true identity reside? At what point in the boot process or on a running system does that get decided? How can I eradicate one of these identities entirely? What is the role of /etc/hosts? Could I have it both ways and have both entries in there? If so, would the IP address for dsl-blah-blah-blah need to be the loopback address 127.0.0.1, or my IP address from Telocity? For the record, the output from **uname -n** (today, at least) is localhost.localdomain. And here's my /etc/hosts file:

```
[root@localhost splus]#  cat /etc/hosts 172.16.87.2
windoze.localdomain  windoze 127.0.0.1  localhost.localdomain
localhost
```

Thanks very much! —Jennifer Bryan, jennybryan@telocity.com

Red Hat keeps the machine name in three places: /etc/HOSTNAME (which is largely ignored; it's for compatibility with Slackware), /etc/sysconfig/network (HOSTNAME and DOMAINNAME) or /etc/hosts. You can change your machine's host name from the shell prompt with **hostname**:

```
moremagic:~# hostname foo
moremagic:~# hostname
foo
moremagic:~# bash
foo:~# exit
exit
moremagic:~# hostname moremagic.merlins.org
moremagic:~# hostname
moremagic.merlins.org
```

—Marc Merlin, marc_bts@valinux.com

Actually, your machine is “localhost.localdomain” when the Ethernet interface is down (ie the system is not connected to the network). Once the eth0 interface is up, the name of your system is “dsl-216-227-xxx-xxx.telocity.com”. According to the description of the problem, I think you don't use a fixed IP address, so your IP (and so your name) will change every time you reboot the system. If your license is configured for a static IP address, it would work only when your dynamic address matches the license address. I don't know a lot about your license manager, but maybe it's possible to use the Linux “dummy” net driver configured for the address defined in the license manager. Here is an example of dummy0 configuration on my system (assuming 192.168.3.3 is the static address used by license manager):

```
ifconfig dummy0 192.168.3.3 up
route route add 192.168.3.3 dummy0
```

If connected to the Net, your system will reply both to the dynamic DSL IP address and the static dummy IP address. If not connected, the system will reply only to the dummy IP address used by the license manager. Another solution is to get a static DSL IP address from your ISP. It's more expensive, but will work every time. I don't think /etc/hosts could help you in case of dynamic IP. The file is useful for adding hosts name+address or name aliases you don't want to add to your DNS. —Pierre Ficheux, pficheux@com1.fr

Floppy Disk Question

Okay, this is silly I think, but I can't find ANY info on it. My floppy worked until today; now when I click on it or try to mount it or try to access a file on it, or format it or put a file on it, I get an error message saying “mount: can't mount, dev/fd0 has wrong major or minor number”. What is this about? How can I fix it? —Sean Lafreniere, sblafren@easystreet.com

Here are the correct permissions and major/minor numbers for /dev/fd0 on Red Hat:

```
   brw-------   1 root   floppy   2,   0 May  5  1998 /dev/fd0
```

If they don't match, you can recreate them with **mknod /dev/fd0 b 2 0**. —Marc Merlin, marc_bts@valinux.com

System Lock

Is it possible for a mouse to cause a system lock? When I am in an X session and I use my mouse (I've tried the Microsoft Intellimouse, and the Logitech First Mouse), sometimes my system locks up and I can't exit the session even forcefully. The mice I've been trying function normally otherwise, and I've tried using them in both PS/2 and serial ports with both the generic and specific drivers. This lock happens only when I use the mouse, but it doesn't happen immediately—it can happen after five minutes of use or 45 minutes of use. There doesn't seem to be a pattern. The techs at Red Hat didn't have a clue, and I'm getting pretty desperate. Any help would be greatly appreciated. —Mike, shirleymg@netscape.net

Considering that a PS/2 mouse uses interrupt 12, which no self-respecting motherboard should assign to anything else, unless you have an ISA card that uses that interrupt, it's probably not a mouse conflict. It's not necessarily the mouse; it could simply be your X server or your graphics card. Try to see if upgrading X works, and if that still doesn't help, try swapping the graphics card with another type. —Marc Merlin, marc_bts@valinux.com

If this happens only with PS/2 mice, then it may be due to a problem in some versions of the kernel driver. A PS/2 mouse is managed by the keyboard controller, so if you lose one, you lose both. Try upgrading to the latest version of the kernel release you are using. Since you report the problem with any kind of mouse, I don't think it's related to mouse activity. Please try to resort to the “Magic SysRq” feature to print out some system information when the lock happens. Using the magic SysRq feature, you should also be able to kill X; if not, you should reproduce the problem in text mode. If no information can be extracted, then the system is locked hard, and it looks more like a hardware problem than a software one. Maybe the processor shuts down for overheating or something similar. —Alessandro Rubini, rubini@linux.it

Text-Based Installation

How can I do an install with Corel 1.1 in text-based mode? Or, rephrasing the question: I'm trying to install Corel 1.1 on a laptop. Since Corel is GUI-based from the get-go, I get an unreadable screen during the install. I was able to do a Red Hat 6.2 install on the laptop, since I was able to put it into text mode during the install. Any words of wisdom on Linux laptop installs? —Ed, Ed.Werzyn@POBox.com

I think Corel requires a Vesa 2.0-compliant video card to install and doesn't offer a text mode install. If you really want to install Corel nonetheless, you can install it onto another machine, boot from an NFS+network+pcmcia boot floppy or CD (http://www.toms.net/rb/ for instance) and do an NFS copy of your installed distribution over to your laptop. —Marc Merlin, marc_bts@valinux.com

Invisible Typing

Hi, I seem to be having a slight problem. When I ctrl-alt F1-F6 from X or quit out of the wm, I am greeted with a blank console; everything I type on that console is blindly written. This happens anytime I go into X and back out to a console. I can issue a reset and I will get the console back, but soon as I go back to X and then back out, it's again blacked out (or no fonts/text showing). I have edited / etc/inittab for the alt + uparrow to issue a reset, but this is a workaround and very annoying. Does anyone know what is actually screwed up and how to fix? I recently compiled X 4.01 and it was fine up until the time I rebooted a day or so later for a kernel compile. I have seen this problem before, so it's not an X 4.01-specific problem. It's an “I screwed something up” problem, I am guessing. I am not running SVGAtextmode, nor are the fonts on console scrambled, they are just blacked out. —Steve Udell, hettar@home.com

If you aren't running your text mode in 80x25, try doing that. Some video drivers don't restore nondefault text modes very well, let alone deal with VesaFB (the graphical text console with a penguin boot logo). —Marc Merlin, marc_bts@valinux.com

Archive Index Issue Table of Contents

Advanced search

New Products

Ellen M. Dahl

Issue #79, November 2000

Oracle PL/SQL CD Bookshelf, EtherLite for Linux and more.

Oracle PL/SQL CD Bookshelf

This CD-ROM contains the complete text of seven books: *Oracle PL/SQL Programming, Advanced PL/SQL Programming, Oracle Web Applications, Oracle Built-in Packages, Oracle PL/SQL Language Pocket Reference, Oracle Built-in Pocket Reference* and *Oracle PL/SQL Programming: A Guide to Oracle 8i Features*. A bonus hard-copy book, *Oracle PL/SQL Programming: A Guide to Oracle 8i Features*, is also included. This CD-ROM is aimed at Oracle PL/SQL developers and is readable with a web browser. The books are fully searchable and cross-referenced.

Contact: O'Reilly & Associates, Inc., 101 Morris St., Sebastopol, CA 95472, 800-998-9938 (toll-free), 707-829-0104 (fax), order@oreilly.com, http://www.oreilly.com/catalog/oraclecdbs/.

EtherLite for Linux

Linux drivers are now available for Digi International's EtherLite line of network serial concentrators. Digi EtherLite serial concentrators simplify the process of adding RS-232, RS-422 and RS-485 serial ports to the network by combining the control and performance of local ports with the convenience of an Ethernet connection. Traffic from all 2, 8, 16 or 32 EtherLite ports is serviced by a single TCP/IP session. EtherLite serial ports appear as local TTYs under UNIX and as native COM ports under Windows NT. These locally administered, “hardware-style” ports allow greater control than standard serial concentrator ports. The Linux driver for EtherLite runs on kernel version 2.2 or higher and is supported on the following, or newer, distributions: Red Hat 6.0, Caldera OpenLinux 2.3 and SuSE Linux 6.3.

Contact: Digi International, Inc., 11001 Bren Road East, Minnetonka, MN 55343, 800-344-4273 (toll-free), 612-912-4952 (fax), info@digi.com, http://www.digi.com/.

SuSE Linux 7.0, Personal and Professional Versions

SuSE Linux 7.0 Personal comes on three CD-ROMs with the core OS and multiple applications; StarOffice 5.2; and three entry-level manuals. Installation is automatic with the YaST2 setup tool; peripheral devices and Internet access configuration via modem and ISDN are carried out with a few simple key entries. SuSE Linux 7.0 Professional is a comprehensive collection of over 1,500 updated Linux tools and software packages for home users as well as IT professionals. Six CD-ROMs or one DVD provide software packages. Features enhanced raw device support and 4GB main memory addressing. All tools necessary to enable the use of Web, proxy, mail and news servers under Linux are included, along with three manuals, plus the SuSE Linux manual.

Contact: SuSE, Schanzäckerstr. 10, D-90443 Nürnberg, Germany, +49-911-7405339, +49-911-74053479 (fax), info@suse.de, www.suse.de/en.

Storm Linux 2000 Starter Edition, Deluxe Edition and Storm Firewall

Bundled in the Starter Edition are the essentials to get users up and running in Linux: PartitionMagic Linux Prep Tool by Powerquest, Netscape Communicator 4.73 and Sun StarOffice 5.2 productivity suite. The Starter Edition is intended for people who want to access the power of Linux without an abundance of third-party software.

The Storm Linux 2000 Deluxe Edition is designed for customers who want a robust Linux system along with bundled third-party software. The Deluxe Edition is based on Stormix's high-performance operating system built around Debian GNU/Linux 2.2 “potato” release. The retail box includes five CDs containing over 4,000 Debian packages, as well as 60-day phone and 90-day e-mail installation support.

Storm Firewall is a flexible, scalable network security solution targeted at SOHO users. It is based on the Storm Linux 2000 operating system and has also been tested to work with the Red Hat 6.x series and Debian GNU/Linux 2.2.

Contact: Stormix Technologies Inc., 555 West Hastings Street, Suite 2040, Vancouver BC, V6B 4N6, Canada, 604-688-9137, 604-688-7317 (fax), http://www.stormix.com/.

HA Web Server Solution for Apache

The SteelEye High Availability (HA) Web Server solution for Apache combines SteelEye's LifeKeeper clustering software for Linux with web solutions. The bundled product enables service providers and IT professionals to create highly available web servers providing access to e-business systems. Features include active/active configuration; centralized Java GUI; support for both shared and local disk storage configurations; data consistency maintenance through an n-way data replication facility; multiple fault-detection mechanisms; shared disk access and seamless failover. The HA Web Server Solution for Apache includes LifeKeeper for Linux 3.0, Apache Web Server Application Recovery Kit, and 24 x 7 support for one year. It is available on Red Hat 6.1 and Caldera eServer 2.3.

Contact: SteelEye Technology Inc.,http://www.steeleye.com/.

Slam v2.2.6

Stabie-Soft released Slam v2.2.6, a family of integrated circuit layout tools. The latest version adds DRC and LVS options to the existing mask layout viewer, editor and delay integrator products. Slam interfaces include GDSII, Spice, Verilog and SDF. The Tcl/Tk-based system provides user extensibility and includes Tcl code to support major third-party verification packages. Contact: Stabie-Soft, 5828 Gentle Breeze Terr., Austin, TX 78731, 512-825-8914, sales@stabie-soft.com, http://www.stabie-soft.com/.

Sculptor

Chilliware, Inc. introduced Sculptor, the first Linux desktop publishing program. Designed to be easier to use than comparable Windows software, Sculptor includes a comprehensive feature set with the latest options in demand by desktop publishing users for creating banners, cards, flyers and more.

Contact: Chilliware, Inc., 3550 Wilshire Blvd., 18th Floor, Los Angeles, CA 90010, 213-365-8700, 213-365-1150 (fax), info@Chilliware.net, http://www.chilliware.net/.

Dial-out Modem Pooling Client

Perle Systems added a Dial-out Client for Linux to its line of 833 Access Servers. The new client allows Linux-based users to share modems with other Linux, Windows 95/98, NT and 2000 workstations on their network by re-routing data traffic from Linux applications normally destined for the COM port to the server modem pool. With this software, Perle 833 Access Servers eliminate the need to attach individual modems on each Linux workstation in the office where dial-out facilities are needed. This reduces the number of PBX ports used for modem dial-out. The new Dial-out client for Linux is supported on Perle 833 v5.64 and newer; Perle 833IS v6.06, 7.0 and newer; and Perle 833AS v6.15 and newer. It is available as a free software upgrade from Perle's web site.

Contact: Perle Systems Inc., 800-337-3753 (toll-free), info@perle.com, http://www.perle.com/.

LinuxMagic VPN Firewall

LinuxMagic VPN Firewall software allows professionals to log in to their office computers from home or connect two offices via the latest encryption technologies. It has built-in 128-bit encryption, IPSEC-compatible VPN standards and LinuxMagic's own NO-WRITE design.

Contact: Wizard Internet Services Ltd., 13595 King George Hwy., Surrey, BC, V3T 2V1, Canada, 604-589-0037, 604-584-0010 (fax), sales@wizard.ca, http://www.linuxmagic.com/, http://www.wizard.ca/.

IRIS-Planning v4.2

Bitbybit Information Systems introduced version 4.2 of its comprehensive timetabling and reservation system, IRIS-Planning. Enhancements include extensions to the workload module and new topology and e-mail modules. It supports reservations by means of the Web, and can generate timetables in HTML format.

Contact: Bitbybit Information Systems, Kluyverweg 2a, 2629 HT Delft, The Netherlands, +31-15-2682569, +31-15-2682530 (fax), info@bitbybit-is.com, www.IRIS-Planning.com.

Immunix Workgroup Server Appliance

WireX announced its latest configuration: Immunix Workgroup Server Appliance. This easy-to-administer, inexpensive, all-in-one software integrates web, e-mail, file and print server functionality, as well as the ability to protect servers from hacker attacks. The Immunix Workgroup Server Appliance includes web-based remote network administration, increasing productivity by providing an intuitive, menu-driven interface that scales to technical and non-technical server administrators. Other features include a remote network user interface; scalable user interface and built-in localization engine. It runs on most off-the-shelf PC hardware. The Immunix Workgroup Server Appliance is based on Immunix OS, a standard Linux distribution which includes integrated server components developed by WireX.

Contact: WireX Communications, Inc., 920 SW 3rd Ave., Portland, OR 97204, 503-222-9660, 503-241-5682 (fax), info@wirex.com, http://www.wirex.com/.

ATI2.5-Inch IDE

BiTMICRO Networks introduced its 2.5-inch IDE entry-level E-Disk ATI25. The solid state flash disk has sustained read/write transfer rates of 4.5MB per second and burst read/write transfer rates of 5MB per second. It features storage capacities ranging from 128MB up to 4.3GB. The ATI25 can operate within the temperature range of -40 to +85 degrees Centigrade and withstand 1,000 Gs of shock and 16.5 Gs (rms) of vibration. The new ATI25 is a drop-in replacement for any standard IDE 2.5-inch hard disk drives, flash disks or solid state disk drives.

Contact: BiTMICRO NETWORKS, Inc., 45550 Northport Loop East, Fremont, CA 94538-6481, 510-623-2341, 510-623-2342 (fax), info@bitmicro.com, http://www.bitmicro.com/.

Atipa Firewall and Firewall Plus

The high-performance Atipa Firewall Plus is a commercial-grade ICSA-rated product that leverages the benefits of Progressive Systems' Phoenix Adaptive Firewall Technology (AFT) and Secure Management System (SMS) and includes additional anti-attack features, such as a De-Militarized Zone (DMZ). The new Atipa Firewall and Firewall Plus are platform-neutral and fully upgradeable. Both feature an Intel x86-based processor with 64MB RAM, 10.2GB HD; 1U rackmount or mid-tower; a web-based software interface and remote administration; DHCP server; IP-masquerading; and an easy-to-use GUI.

Contact: Atipa Linux Solutions, 4700 Belleview, Suite 300, Kansas City, MO 64112, 800-360-4346 (toll-free), 816-595-3001, sales@atipa.com, http://www.atipa.com/.

Archive Index Issue Table of Contents

Advanced search