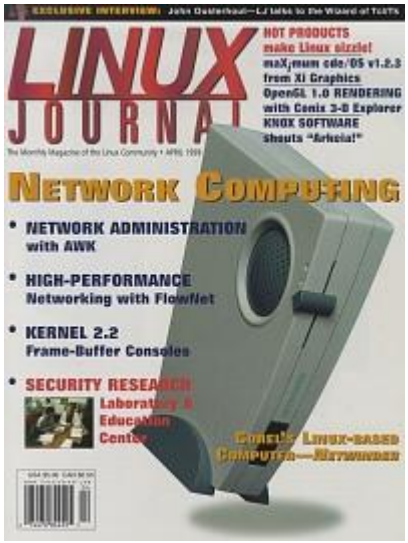
Linux Journal Issue #60/April 1999



Reviews

Columns

Departments

Strictly On-line

This paper is a general presentation of the Xxl project and of its latest version, describing the choices that drove the design of Xxl and its main charactertistics.

Network Programming with Perl

Using Perl to make network task is easy—here's how.

Linux in Enterprise Network Management

Providing Network information to customers on an intranet saves both time and money for this international chemical company.

Alphabet Soup: The Internationalization of Linux, Part 2

Mr. Turnbull takes a look at the problems faced with different character sets and the need for standardization.

Archive Index

Advanced search

From the Editor

Marjorie Richardson

Issue #60, April 1999

Linux's biggest strengths has always been as an operating system for all types of servers: e-mail, web, boot, you name it. This makes it ideal for use in networking computers.

Network computing has been a hot topic for some time now, so it should come as no surprise that we included it in our 1999 editorial calendar. Network computers provide Internet/intranet services at a fraction of the cost of a PC. With the popularity of the World Wide Web, the network computer's time has come—companies dealing in network computers show high performance on the New York Stock Exchange.

One of Linux's biggest strengths has always been as an operating system for all types of servers: e-mail, web, boot, you name it. This makes it ideal for use in networking computers. IGEL's Etherminal proved Linux's worth as a thin client back in 1994 and has kept on proving it ever since. Last year, Corel's NetWinder joined the ranks.

Whether you want to learn more about using Linux in your network, how to easily access your network services or how to speed up your network, this month we have the articles you need. In addition to the features listed below, we also have two articles on our web site (see “Strictly On-line” in the Table of Contents): one explaining the DECnet protocol and one providing Perl scripts for accessing your network. Also on the web is an article about how one international company is using Linux for network management.

Along with these articles on the web site is the second part of last month's internationalization feature by Stephen Turnbull, “Alphabet Soup”. In this follow-up, he discusses standardization of character sets—don't miss it.

“Strictly On-line” articles can be found at http://www.linuxjournal.com/issue60/.

Hardware News

First it was Netscape then the major database providers, and now major hardware vendors are jumping on the Linux bandwagon. At the end of January, Hewlett-Packard Co. and Silicon Graphics, Inc. both announced they would be providing Linux as an option on their computers with Intel chips. Rumor has it that Compaq, IBM and Dell will be following suit. Even Apple has said they will make Linux an option. These companies are not doing this to become popular with the open-software crowd—they are doing it to make money. They have seen that a market clearly exists and are taking advantage of that fact. More on making money with open-source software will be found in our June *Enterprise Solutions* supplement.

Marjorie Richardson, Editor

Featured Articles

Archive Index Issue Table of Contents

Advanced search

Product Review: Corel's NetWinder

Marcel Gagné

Issue #60, April 1999

A review of this networking computer from Canada.

- Manufacturer: Corel Computer
- URL: http//www.corelcomputer.com/
- Price: Varies with model, see web site
- Reviewer: Marcel Gagné

Corel's NetWinder has to be one of the coolest computers I've ever seen. The slick little grey box makes me think of those cream-filled half-moon cakes with a little chunk taken off the edges. Sitting in its green plastic base, it beckons with a single deep red window up front and two status lights telling you everything is fine. It is about the size of my notebook, but unlike my notebook, it weighs next to nothing.

My time with the NetWinder was up a few days ago, but I have been hanging on to it due to problems in scheduling a drop-off time with the Corel sales representative. On my first attempt to deliver the unit, I stopped off at a customer's site, NetWinder in hand. It took an hour before I was allowed to leave again, as everybody in the office had to get a look. “That's a computer?” was heard time and again, as was “And it has *what* inside?” and “Tell me again what it can do?” If nothing else, the NetWinder gets looks. Luckily, there's more here than just looks.

Figure 1. The NetWinder fits into any decor!

Specs

The NetWinder is based on a 275MHz StrongARM SA-110 RISC processor which delivers 250 MIPS. It comes with 32 or 64MB of RAM and a 2, 4 or 6GB disk. There are two Ethernet interfaces, one 10 and one 10/100 fast Ethernet port. It draws 15 watts from its power brick, about the equivalent of a couple of night lights. The NetWinder WS also comes loaded with Perl for CGI scripting, the Apache web server, FTP, TELNET and DNS services. Also included are multimedia support, a 16-bit stereo sound card and 2MB SVGA/XVGA video.

The unit I reviewed is actually a DM demo. That translates to “a little bit of WS, DM, GS and LC all rolled into one”. Its OS version is based on Red Hat 4.2 with some Corel extensions. By the time you read this review, NetWinder will come preloaded with Red Hat 5.1.

A few different configurations are available. At this moment, there are five NetWinder configurations. The WS (Web Server) and the DM (Development Machine) are currently shipping. The LC (Linux Computer) should be out by the time you read this; I was told “in time for Christmas 1998”. The other two models, the RM (Rack Mount) and GS (Group Server), are expected in early 1999. Pricing and configuration options are available from the Corel Computer web site at http://www.corelcomputer.com/. The open source software and updates are available at http://www.netwinder.org/.

The Roundup

The WS is designed to be an “out of the box” web server. Small ISPs and corporate intranets or web sites are the most likely candidates for the WS.

The DM is aimed at the software designer and comes with a suite of development and programming tools. It is primarily for Corel NetWinder project developers.

The LC is seen as a business or enterprise desktop client, although I also see this as potentially quite attractive to the home user looking for a robust alternative to running MS Windows. It comes with X, the KDE windows environment, Netscape Communicator and WordPerfect 8. Supporting many client/server models, the LC comes with Sun's Java VM, a Citrix ICA client to access 32-bit MS Windows applications and good old terminal access.

For the workgroup environment, Corel offers the GS. Designed as a local enterprise server, the GS offers e-mail services, HTML authoring, discussion groups, and document management and indexing functionality.

Finally, there's the RM. This NetWinder will appeal to larger ISPs. Each RM can accommodate two NetWinders in the space of a single rack unit. That means a possible 80 NetWinders in a standard rack. Each RM is designed to be hot swappable from its partner. One from each pair can be removed without powering down the other unit.

First Impressions

I decided to review this unit while leaning toward the WS model, a more likely choice for ISPs and corporate intranets. When I got my NetWinder home, I plugged the unit in, hooked up the included mouse and keyboard and plugged in a 15-inch Digital SVGA monitor. The NetWinder does not come with a monitor, because in the case of the WS, Corel imagines the unit running in the background without being looked at. After all, it *is* designed to be a server and can be administered through a web interface. Personally, I can imagine lots of people wanting one of these on their desktops.

After a few seconds, I heard little start-up tones, “Too-loo-toong!”, followed by a strange, childlike voice saying, “Welcome to NetWinder”. (I later found out that this is one of the developer's children. The voice is kind of cute.) Next, I logged in, as per the tiny (not much more than a dozen pages) user's manual that came with it, as I was anxious to get started. I completely ignored the part about “setting it up for your network”.

I typed **startx** and was greeted by a bright, beautiful KDE desktop with Corel's logo floating on a black background. I flipped through the four virtual desktops, each with a different background. Number three, with its stone wall image as the background, nearly blew me away; the brilliant definition begged me to reach out and touch it in order to convince myself it wasn't real. Corel has put nice video into their NetWinders. This was actually the first time I had seen the KDE desktop, and I was so impressed with it that I now run it on my Dell XPi notebook. KDE is not included on the WS, but I couldn't resist playing with all the toys on my DM.

I wandered around the desktop, played a couple of hands of poker, and finally dragged myself back to writing this review. To set the machine up for my network, I plugged the 10Mbps port into a free port on the hub, typed **netconfig** and modified the address to fit in to my 192.168 existing class C. Then, I used Netscape to access the NetWinder from my Dell notebook running Red Hat 5.2.

Figure 2. So simple, even Natika can use it.

WebFront & Administration

Right away, I was presented with WebFront, Corel's administrative interface for the WS. It took me awhile to figure out how to log in to the administration utility. After chasing through some of the CGI scripts under the /admin directory, I found a reference to /usr/bin/htpasswd.sh which apparently you must run at least once in order to get WebFront to realize your admin user is legit. This script creates the .htpasswd file.

WebFront is a browser-based tool that makes it possible to administer your NetWinder through a few simple forms. You can add, delete or modify users and groups through the interface. The clean, simple interface also serves up statistics on your NetWinder's performance in a graphical form, giving you glimpses of your web site's activity over different time scales.

To get a feel for its performance, I uploaded a few web sites that were done in Microsoft FrontPage. The transfer was very fast, a testament to the NetWinder's zippy little combination of processor and network architecture. I did a couple of these, one for our SF magazine *TransVersions* and another for a customer's intranet. Both sailed by faster than the “time to go” in FrontPage could keep up. What it estimated would take three plus minutes flew past in less than ten seconds.

Access to the pages was very fast. I fired up Netscape and as fast as I could click links, pages appeared in my browser. It was almost disconcerting. My local

Pentium-powered Linux system, which we use as our corporate gateway/e-mail server/development machine, seemed to crawl in comparison. I was starting to get a touch of server envy.

Corel benchmarks with 16 clients connected to the NetWinder at 70 requests per second or nearly 350,000 bytes per second. You can expect as many as 150 concurrent requests per second, depending on client load.

Figure 3. The author and his NetWinder

A Few Bumps in the Road

My experience wasn't all roses. One of the things my company does regularly is set up Internet gateway, e-mail server, firewall, dial-on-demand combination systems that give customers inexpensive access to the Internet. We use PPP and masquerading to quickly link up a small office for transparent access. I decided to try this with the NetWinder.

Using the supplied netconfig, I found no option for modifying the PPP interface, so I created one using the Red Hat **netcfg** utility and tried to activate it. A message to the console informed me, “Sorry—this system lacks PPP kernel support”. No problem, I thought. I'm a UNIX guy. I can recompile the kernel with PPP support and be on my way. It turns out I was mistaken. There is no kernel source on the system, so PPP is impossible at this time.

Another thing I found disconcerting was the lack of an “off” switch. Call me foolish, call me irresponsible, but I find the fact that you can't shut down the machine distressing. While I realize you'll want your network gateway to stay up forever, I'm used to powering down my system every once in a while. If you want to shut down the NetWinder, you can use the standard **shutdown** command and wait for the “System halted” prompt before disconnecting power by pulling the power cord. That's right. You disconnect power, rather than turning it off. Even then, the FAQ from the netwinder.org site says you should

not power down. The NetWinder has no battery for backup of its system clock. What it does have is a “big capacitor” that will hold a charge for about two days —any longer and you may need to reset your clock. When I got my NetWinder for evaluation, it had been off for some time and seemed to think we were already in the next millennium.

One other thing I will admit is fairly minor. I find the fan a bit loud, which is silly since the NetWinder is still quieter than my usual server. Without the hum of the power supply, the disk churning away and all the other noises one usually expects from their computer, the NetWinder's little fan seems, well, loud—*bizarre*.

A little later, I wrote up a small list of “would-be-nices” to pass along to Corel. I'd like some kind of database. The freeware Postgres would be enough to make me happy. I'd also like that PPP support, a built-in fax modem *and* the source code so I could hack my kernel when I felt the need to.

Corel Responds

The next day, I called Corel's tech support engineers and discussed this. They told me that in early 1999, NetWinder would have PPP support and should include a V.90 modem for access. This pleased me very much. Right now, the NetWinder is designed to sit behind a router with one interface configured for the internal network and the other (most likely the 10 Mbps interface) hooked up to the outside world. This makes it a winner with small ISPs and corporate intranets.

The Corel engineers have already considered many of these things. After chatting with them, I was told that future NetWinders would include full Red Hat distributions which should take care of most of my software concerns. They have some other cool ideas as well. Corel intends to offer customized NetWinders using swappable daughter cards; some would come with ultra-wide SCSI to plug into RAIDs, others with ISDN. Corel is keeping an open mind.

The LC was out at the end of 1998. The WS and DM are available now. Corel is also looking into a TC or thin client. It would run a core Linux OS and take everything else from the network. Because it will be essentially diskless (OS on board), this NetWinder will be completely silent; it wouldn't even need a fan.

Ah, yes, the fan—Corel also thinks the fan is loud and is currently working with different designs to make things quieter.

Final Impressions

The NetWinder seems like a good bet and I'm going to keep a close eye on how this product plays out. What you need to do is decide which NetWinder will suit your needs. In order to satisfy my current customer's needs, I'd most likely turn to the GS when it becomes available.

The NetWinder is small and fast and its power consumption is minimal. It looks slick. Best of all, NetWinder runs on Linux.

Marcel Gagné lives in the mythical city of Mississauga, Ontario. Besides being a space alien, adventurer, pilot, magician and international man of mystery, he is president of Salmar Consulting Inc., a systems integration and network consulting firm. He also writes science fiction and fantasy, and edits TransVersions, a science fiction, fantasy and horror magazine. He has loved UNIX and all its flavors for over 14 years now and will even admit it in public. He can be reached via e-mail at mggagne@salmar.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

FlowNet: An Inexpensive High-Performance Network

Erann Gat

Mike Ciholas

Issue #60, April 1999

A look at current state-of-the-art network hardware and protocols with a solution for the slow network problem.

We have been using Linux to develop a new high-speed network we call FlowNet. This project has been a “virtual garage” operation, involving only two people, one in California and the other (at various times) in Massachusetts, Pennsylvania and Indiana. We transferred designs and code over the Internet and hardware via Federal Express. The result is a unique network that combines the best features of today's current standards into a single design. FlowNet is currently the world's fastest computer network capable of operating over 100 meters of standard category-5 copper cable. The software for FlowNet was developed and currently runs exclusively under Linux.

To appreciate how FlowNet works, it is important to understand some details about network hardware, so we will start with a brief tutorial on the current network state of the art.

Network Background

The dominant hardware standard for local area networks today is Ethernet, which comes in dozens of variants. The only feature common to all forms of Ethernet is its frame format; that is, the format of the data handled directly by the Ethernet hardware. An Ethernet frame is a variable-size frame ranging from 64 to 1514 bytes, with a 14-byte header. The header contains only three fields: the address of the sender of the frame, the address of the receiver and the frame type.

Shared-Media Ethernet

Ethernet design has two major variations called shared-media and switched. In shared-media Ethernet, all the network nodes are connected to a single piece of wire, so only one node can transmit data at any one time. Ethernet uses a protocol called carrier-sense-multiple-access with collision detection (CSMA/CD) to choose which node is allowed to transmit at any given time. CSMA/CD is a non-deterministic protocol and does not guarantee fair access. In fact, in a heavily congested network, CSMA/CD tends to favor a single node to the exclusion of others, a phenomenon known as the capture effect. Being on the wrong end of the capture effect is one way a network connection can be lost for a long period of time.

The CSMA/CD protocol does not allow a node to start transmitting while the wire is being used by another node (that is the carrier-sense part). However, it is possible for two nodes to start transmitting at almost the same time. The result is that the two transmissions interfere with each other and neither transmission can be properly received. The period during which a collision can occur is the time from when a node starts to transmit to when the signal actually arrives at all other nodes on the wire. This time depends on the physical distance between the furthest nodes on the wire. If this distance is too long, a node might finish transmitting a frame before it arrives at all nodes on the wire. This would make it possible for a collision to occur that the transmitting node would not detect. In order to prevent this from happening, the physical span of a shared-media Ethernet network is limited. This distance is known as the collision diameter; it is a function of the time necessary to transmit the shortest possible Ethernet frame (64 bytes). The collision diameter of a traditional Ethernet operating at 10Mbps is about two kilometers, which is plenty for most local area networks. However, the collision diameter shrinks at faster data rates, since the time it takes to transmit a frame is less. The collision diameter for Fast Ethernet, which operates at 100Mbps, is 200 meters—a limit that can be constraining in a large building. (The collision diameter for Gigabit Ethernet would be 20 meters, but because this distance is so ridiculously short, Gigabit Ethernet does not use CSMA/CD.)

Switched Ethernet

The way to get around the limitations of shared-media Ethernet is to use a device called a switch. A switch has a number of connections or ports, each of which can receive a frame simultaneously with the others. Thus, in a switched network, multiple nodes can transmit at the same time. In a purely switched network, every node has its own switch port and there can be no collisions. However, there can still be resource contention because it is now possible for two nodes to simultaneously transmit frames destined for a single node, which still can receive only one frame at a time. The switch must therefore decide

which frame to deliver first and what to do with the other frame while waiting. Switches typically include some buffering so that contention of this sort does not necessarily result in lost data, but under heavy use, all switched networks will eventually be forced to discard some frames.

How does the switch decide which frames to drop? Most switches simply operate on a first-in/first-out basis. That is, when they are forced to drop frames, they drop the most recently received ones. Not much in the way of alternatives is offered because no information is in the Ethernet header to indicate which frames are less important and should be dropped first. As a result, when most switches become congested, they drop frames essentially at random.

That behavior creates a serious problem. The response of most network protocols, including TCP/IP, to dropped frames is to retransmit the dropped frames. Thus, network congestion leads to randomly dropped frames, which leads to retransmission, which leads to more network congestion, which leads to more randomly dropped frames. When this happens, many networks, in particular the Internet, will often come to a screeching halt.

Quality of Service

The only way to solve this problem is to add information to the frame to give switches guidance on how to handle individual frames. For example, if a frame is tagged as part of an e-mail message, a switch would know that it is perfectly acceptable to delay this frame, but also that it should probably not be discarded. On the other hand, if this frame is part of a video stream, then a switch would know that if this frame cannot be transmitted right away, it should be discarded, resulting in a small glitch in the video. Information on how a frame should be handled in a congested network is known as quality-of-service information or QoS.

QoS information can be provided in three ways. The first is to completely redesign the network from the ground up. This is the approach used by the Asynchronous Transfer Mode (ATM) network design. ATM is a circuit-switched rather than a packet-switched network. In a packet-switched network like Ethernet, each data frame contains its destination address in its header. The process of delivering a frame to its destination is similar to that of delivering a letter. At every switch, the destination address is looked up in a table to see where it should go next. Circuit-switched networks like ATM put the destination address into only one frame, called a flow-setup frame or flow-setup cell. The flow-setup cell establishes a route through the network, much like placing a phone call does. Subsequent frames are automatically routed through this pre-established connection. The flow-setup process allows the ATM network to

allocate network resources ahead of time in order to provide quality-of-service guarantees.

ATM's circuit-switched design is fundamentally incompatible with Ethernet's packet-switched design. ATM also differs from Ethernet in the size of its frames. Where Ethernet uses variable-size frames, ATM uses fixed-size 53-byte frames or cells, of which five bytes are header and 48 bytes per frame are payload data. This leads to a serious problem: the rate at which cells must be routed is so fast that it can be done only with custom hardware, which makes ATM very expensive.

The second way to provide QoS information is to put it in the data portion of an Ethernet frame. This is the approach being taken by the Ethernet community, through protocols such as RSVP. The advantage to this approach is that it is backwards compatible with existing hardware, which is important because an enormous Ethernet infrastructure is already installed. ATM can be made to interoperate with Ethernet through a technology called LAN emulation (LANE), but it is both difficult and inefficient.

The problem with implementing QoS using the existing Ethernet frame format is that most existing hardware will not recognize the new protocols associated with QoS. This can undermine the QoS mechanisms by injecting frames into the network which are not properly tagged or by not handling tagged frames properly. Thus, while this approach is backwards compatible with existing hardware, it probably won't be reliable unless most of the existing infrastructure is replaced.

The third approach is to add QoS information to the Ethernet header. This is a non-backwards-compatible change, but not as radical a redesign as ATM, and it can be done in a way that makes it easy to interoperate the new network with existing hardware. This is the approach we have taken in the design of FlowNet.

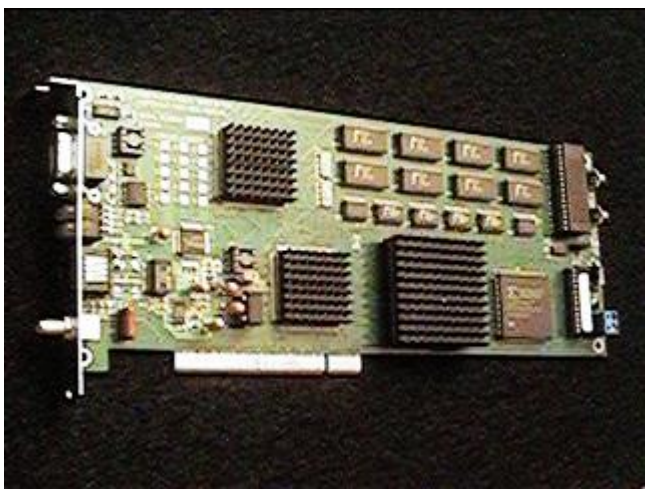


Figure 1. The FlowNet prototype is built entirely from off-the-shelf components and fits in a standard PCI slot just like an Ethernet card. In production, this board could be reduced to just two or three chips.

The FlowNet Architecture

Like ATM, FlowNet is a switched network based on fixed-size cells. Unlike ATM, FlowNet cells are large—800 bytes instead of 53. This allows room for a 14-byte Ethernet header plus an additional QoS extension. The QoS extension header is 18 bytes, making the full FlowNet header 32 bytes long. The remaining 768 bytes (=256+512) are data payload.

FlowNet interoperates with Ethernet through a simple bridge device. To convert a FlowNet cell into an Ethernet frame, the bridge simply strips off the QoS extension. To go the other way, it generates a QoS extension with default or user-configured values. For example, the bridge could be programmed to give frames from certain workstations high priority, while frames from other workstations receive low priority.

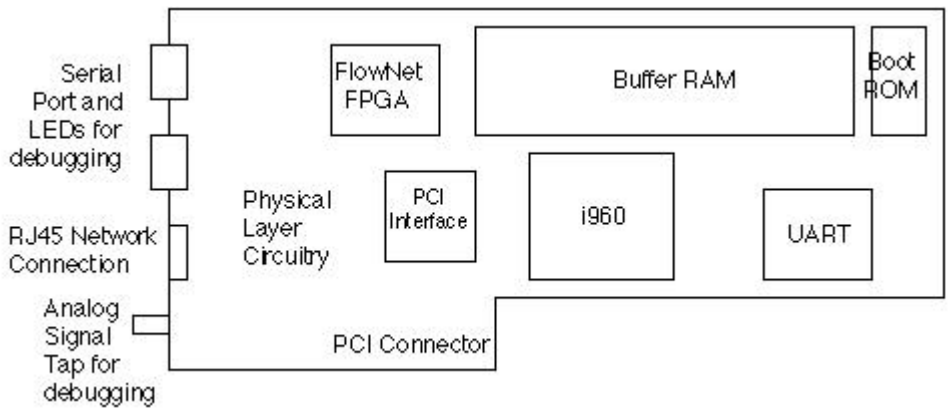


Figure 2. A diagram of the FlowNet prototype showing the location of the major components.

Distributed Switching

The FlowNet architecture is innovative in ways that go beyond the structure of the frame. A FlowNet network interface card (NIC) is quite simple, consisting of a transmitter, receiver, some memory and a microprocessor. NICs are logically daisy-chained together to form a loop. Physically, FlowNet uses a star topology with a hub, just like Ethernet. When a node sends a cell, the cell is received and retransmitted by every node between the transmitter and the receiver, an arrangement known as a store-and-forward loop. To reduce latency, FlowNet uses a technique called cut-through routing, which allows a cell to be retransmitted as soon as the header is received.

The resulting network is a switched network, with a unique feature: it does not require a switch. Instead, each NIC acts like a little two-port switch, with one port on the network and another at the host interface. Switching capability is

distributed among all the nodes on the loop. Cell routing decisions are made in software by the on-board microprocessor, which provides sophisticated quality of service without expensive custom hardware.

Making cell-switching decisions in software is possible due to FlowNet's large cell size. Cell-switching decisions are made on a per-cell basis. Larger cells mean fewer cells for a given data rate, which means fewer cell-switching decisions to be made. Current FlowNet prototypes can switch data at 250Mbps full-duplex (for a total data rate of 500Mbps) using an Intel i960 microprocessor.

Development

FlowNet is a state-of-the-art network. Beside being the fastest network available over 100-meter runs copper cable, it is the only network available that provides quality of service and is efficiently interoperable with Ethernet. FlowNet was developed on a shoestring budget (about $20,000 US for a dozen prototypes) by the authors working alone in their spare time.

Open-source software, including Linux and Intel's gnu960 development tools, was instrumental in allowing this to happen. Linux was used to develop both the on-board firmware and the device drivers for FlowNet. Several Linux features were crucial for allowing us to meet our objectives. The first was the availability of model code for device drivers. Because FlowNet's interface is so similar to Ethernet, we were able to use Donald Becker's Tulip driver as a model and adapt it for FlowNet rather than starting from scratch.

The second Linux feature that helped immeasurably was kernel modules. Because device driver code is kernel code, it was not possible to run it as an application. Without modules, device drivers have to be tested by compiling them into the kernel and rebooting. This adds time to the development cycle. With kernel modules, kernel code can be dynamically linked and unlinked, reducing the testing cycle to less than a minute. We built a kernel module for FlowNet that loaded the card's firmware through the PCI bus during initialization. This made it possible to recompile and restart all the FlowNet software with a single **make** command. As a result, all of the software for FlowNet was developed in less than three months.

The only time rebooting was necessary was when a bug in the driver code caused a kernel panic. Sometimes this would cause the machine to crash, but not always. At no time during the development process did we ever lose any data as the result of a kernel crash, despite the fact that on occasion we were overwriting critical kernel data structures with random bits. Linux is astoundingly robust.

Conclusions

FlowNet would not have come into being without Linux for a development platform. The hardware costs stretched our meager budget to the limit. The development tools needed to develop FlowNet for a commercial OS would have killed the project.

FlowNet was first conceived in 1993. Although Fast Ethernet (and soon, Gigabit Ethernet) seem to be taking over the world, FlowNet is still unique in offering gigabit performance and quality of service without requiring fiber optic cabling or discarding Ethernet infrastructure. Linux made it possible to build FlowNet as a private development—it almost certainly could not have happened any other way. FlowNet is not currently in production; contact the authors for more information (http://www.flownet.com/).

Erann Gat (gat@flownet.com) is a senior member of the technical staff at the Jet Propulsion Laboratory in Pasadena, California.

Mike Ciholas (mikec@flownet.com) is the owner of Cedar Technologies, a hardware design consultancy in Newburgh, Indiana.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Using Linux with Network Computers

Brian Vincent

Issue #60, April 1999

A look at one man's experiences setting up Linux as an application and boot server for Neoware network computers.

For over three years, the computer industry has been touting the benefits of network computers, including higher reliability, lower cost of ownership and simpler administration. Yet despite this, few people have seen a network computer. Even fewer have configured, installed and supported them.

The premise behind network computers is increasing both reliability and manageability. The former is done by removing components with high failure rates, such as the hard drive and fans; the latter is improved by centralizing applications on servers. A standard proposed to the Open Group embodies these principles in the “NC1 Reference Profile”. The hardware specifications require only a keyboard, pointing device, network adapter, audio, 8MB RAM and 640x480 resolution. The PC industry proposed the “NetPC” standard which further requires an X86 style processor, hard drive and plug-and-play compatibility.

The NC1 standard is followed by Neoware's line of network computers. In addition, Neoware added a Java virtual machine, support for PCMCIA cards, higher video resolutions and more memory. Also supported is a wide variety of network protocols, making integration into existing networks fairly easy. For this, Neoware is consistently rated among the top of the NC manufacturers and is also the reason we chose to work with them.

Figure 1. A Look Inside

The company I work for is Unique Systems, Inc., based in Toledo, Ohio. We have a large customer base throughout the Midwest. Our primary focus is custom programming and support for an accounting and job cost system. In addition, we are often asked to perform system administration for our clients. As a result, simplicity and remote administration are essential. We do not have the time or the staff for long-term on-site maintenance. NCs are a perfect fit.

Before I get into configuring and using an NC with Linux, let me describe the basic functionality of a Neoware NC, in particular NeoStation 200. When the NC is first turned on, it establishes a TCP/IP network connection and obtains its IP address via BOOTP, DHCP or from a value stored in flash memory. Next, it downloads its operating system (Neoware uses netOS) from a boot server and initializes it. The user is then presented with a graphical login screen similar to XDM. It has the ability to authenticate against any host on the network. After logging in, a window manager is started. The default is **netoswm**—similar to FVWM95. Open Look and Motif window managers are also available. All the window managers are capable of running remote X applications or Windows applications using ICA. Many applications can be downloaded and executed out of RAM as needed; others can be run remotely using either X or ICA (Independant Computing Architecture) thin-client desktops.

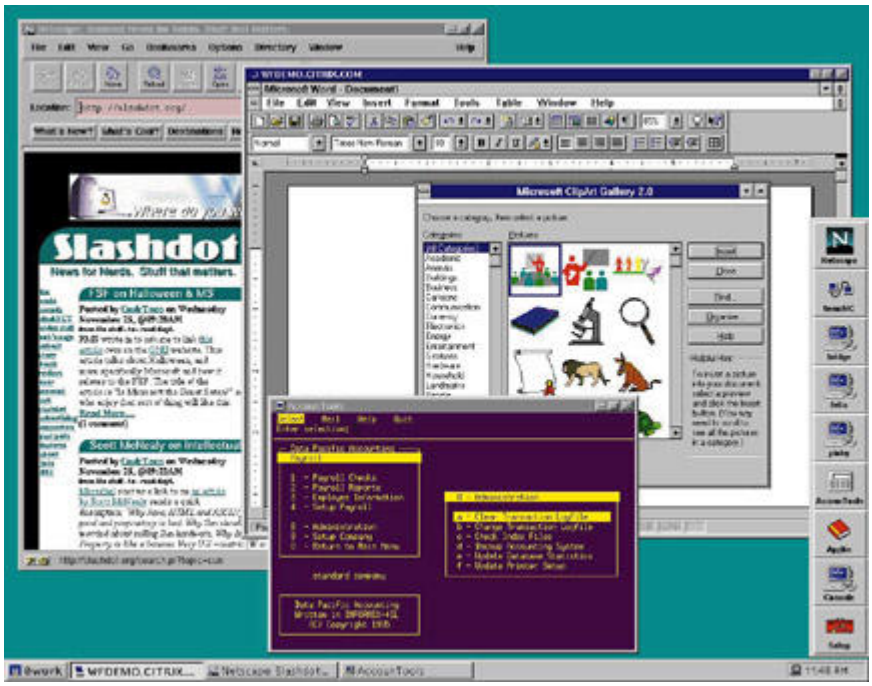


Figure 2. A Windows ICA Session, Remote Character Application and a Netscape Window

Many of our clients need to do only basic office tasks: access to the accounting system (which we can provide via a terminal session or a Java applet front end), e-mail and an office suite. An NC is a good match for these applications. Terminal emulation is provided by the TeemNC application. E-mail can be done with Navio's Netscape Navigator 3.0 for netOS. For an office suite, we install Applixware on a Linux server and execute it remotely from the NC (this is much faster than you might think).

What else do you need beside a Linux server to make it work? The short answer is—nothing. In practice, however, a few things need to be done on other platforms. In particular, I have had difficulty getting the daemon **in.timed** to work properly to provide time services, and several commercial applications have not yet been ported to Linux.

Let's start from the beginning with how the netOS gets loaded onto the NC and what part Linux plays in the process. When a NeoStation is turned on, it gives the option of entering a configuration screen; all the network configuration can be done from here. The IP addresses of the DNS server, gateway and boot host (all Linux machines) can be supplied, as well as the address of the NC itself. The location (directory) on the boot host where the netOS operating system resides also needs to be supplied. The access method can be either NFS, FTP or TFTP, with NFS being the preferred method. Once the NC is configured and restarted, it searches the network for the specified boot host, attaches to it using the chosen protocol and downloads its OS.

Neoware supplies netOS on a CD with an installation shell script. This script ensures the license key is valid, prompts for the installation directory and copies files off the CD into that directory.

Configuration of the boot host is a little more complicated. NFS needs to be configured if it is to be used as the access method. Usually this involves editing the /etc/exports file and adding an entry for the netOS installation directory. You might also want to make sure NFS starts up on boot. We usually configure the boot host to run domain name services and **sendmail** for the network, just because Linux is so good at it.

Now, assuming all this is in place and running, the NC should boot up and present the user with a login screen. Any host can be used to authenticate against, but it is easiest to supply it with a user account on the Linux boot host. After logging in, the NC looks for a window manager configuration to run. It defaults to a system-wide configuration on the boot host if it can't find a custom one in the user's home directory.

Any applets/applications installed with the netOS are part of the default window manager configuration. These include Netscape Navigator 3.0 and miscellaneous utilities such as a calculator. You can run them by navigating through a menu tree similar to the infamous “Start” button, or through a floating button bar on the desktop. Unfortunately, adding anything requires knowledge of how to edit a window manager configuration file. The saving grace is that the format is similar to many “standard” UNIX configuration files. One of the first things I do is edit this file to remove the button for the “Setup” utility and add buttons for Applixware and TELNET access to remote hosts.

Neoware ships an ICA client with netOS that allows you to run Windows applications alongside X, Java and character applications. If you need to run Windows applications, you can set up an NT Terminal Server with Citrix's MetaFrame for Terminals (which is significantly less expensive than other MetaFrame products). The client can be started through the root window menus. In the screenshots (see Figures 2, 3 and 4), you can see all of these running alongside each other. Interestingly, when that screenshot was taken, only half of the system memory was being used—16MB were free.

Working with netOS is not very difficult. The console window on an NC offers several UNIX-like commands that can be used to show the current hardware configuration, connect to remote machines (such as by TELNET and FTP) and browse the file system. The file system is similar to an NFS file system: most of the files reside on a remote computer, but the local ROM can also be examined. Very few configuration files on the server ever need to be edited and almost all

of them lie in the same directory. Devices can be accessed, similar to UNIX. You can access the parallel port through /dev/pp0, for example.

Is it that easy? Yes, provided you have a good understanding of configuring NFS and adding hosts to the network. Almost all the problems encountered when setting up a boot host are associated with NFS permission problems on the part of the NC. A boot log is generated on the NC that can be viewed to troubleshoot problems. Setting up the first NC on the network is the hardest part, because of all the configuration that has to be done on the server. After that, getting other NCs up and running applications on the network happens about as fast as you can unbox them. In fact, it is not an exaggeration to say it takes longer to open up the boxes than it does to configure the NC.

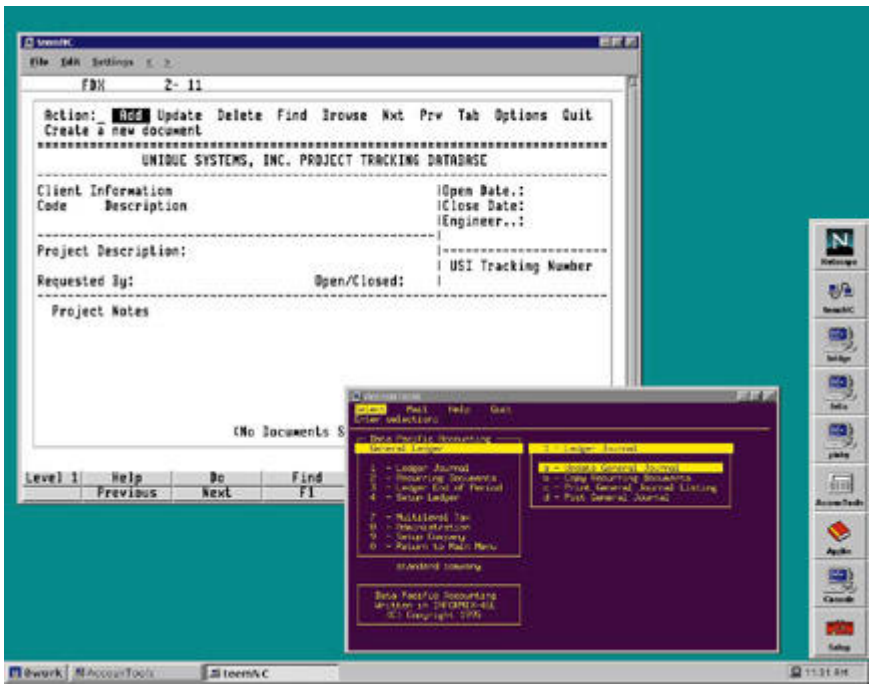


Figure 3. Some Character-based Applications Running in Different Windows with Different Emulations

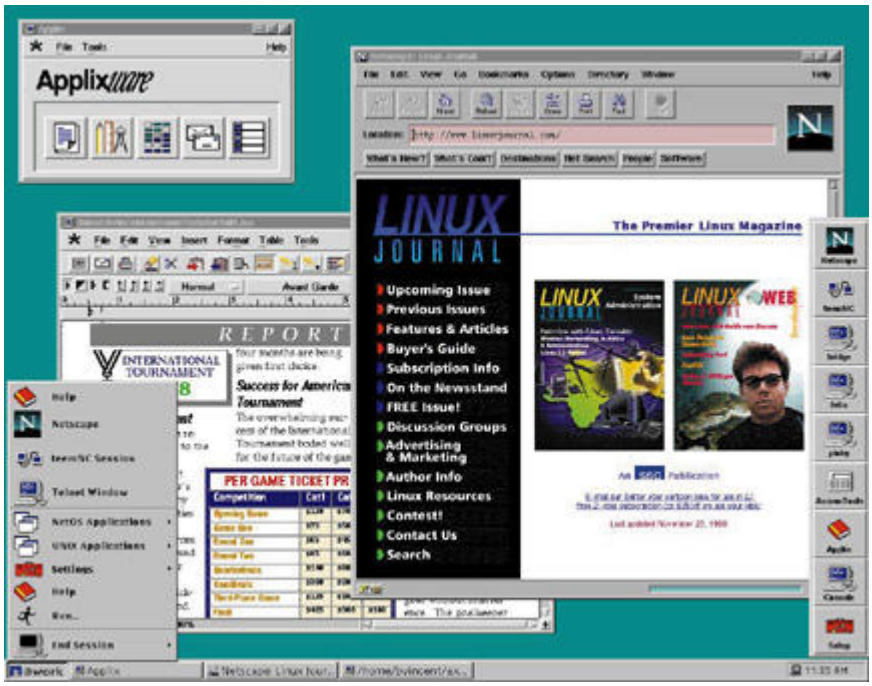


Figure 4. A Few X Applications, Applixware and Netscape

With the amount of X and character-based applications available, why haven't NCs caught on? The most obvious reason is the drop in price of PCs. Before 1995, it was next to impossible to find a PC for less than $1000 US; now, nearly every computer manufacturer offers a few to choose from. Another reason is the lack of Java applications that were promised. NCs were supposed to be the ideal platform to run all the Java applications that have yet to appear. For better or for worse, the industry leaders in the NC market are not the industry leaders in the PC market.

All that aside, network computers are still the ideal platform for many people who use computers. Anyone using a terminal to access a character-based application on a server can benefit enormously by the ability to have multiple windows open and viewable simultaneously on an NC. The learning curve on an NC can be tailored to fit individual users by changing the complexity of the desktop. For curious users or those unfamiliar with computers, it is next to impossible for them to accidentally render an NC unusable. Network computers and Linux servers make a powerful combination.

Brian Vincent (brian@uniqsys.com) is a systems administrator in Toledo, OH. He has been using Linux since 1995 when he discovered he could complete his

course work from home instead of trudging to crowded computer labs. In his spare time he enjoys downhill skiing, backpacking and rock climbing.

Archive Index Issue Table of Contents

Advanced search

Network Administration with AWK

Juergen Kahrs

Issue #60, April 1999

If you are looking for an easy way to access your network services, AWK scripting provides the means.

What does the scripting language AWK have to do with networking? In the May 1996 *LJ*, Ian Gordon introduced us to AWK and demonstrated how to solve common problems with this scripting language that is part of Linux and every UNIX-compatible operating system. He summarized:

> If your main concern is getting a working program written as quickly as possible, you probably do not want to wrestle with C or C++ for a week to perfect the most efficient algorithm. By trading off the speed advantages and control features of C (or another compiled language) for ease of use, gawk lets you get the job done quickly and relatively painlessly.

With this kind of efficiency in mind, it would be nice to also access network services with short AWK scripts. However, standard AWK has no functions for networking, and most AWK users would probably object to the introduction of such functions. AWK should stay the small, simple and powerful language it is now. Release 3.1 of GNU AWK does not introduce special functions for socket access (as Perl and C have), but uses a special file name for it. By treating network connections like files, even novices can write web clients with a few lines of AWK.

Finding Who is Logged In

Let's look at an example. It asks the **finger** service of your local machine if a particular user is logged in.

```
BEGIN {
 NetService = "/inet/tcp/0/localhost/finger"
 print "
 while ((NetService |& getline) > 0)
   print $0
```

```
    close(NetService)
  }
```

Store this script in a file named finger.awk and let GNU AWK 3.1 execute it by typing **gawk -f finger.awk**. The strange pipe symbol, **|&**, is the second and last addition to the AWK language needed for networking. When communicating over a network, we have to use **|&** instead of simply **|**.

After telling the service on the machine which user it is looking for, the program repeatedly reads lines that form the reply. When no more lines are received (because the service has closed the connection), the program closes the socket before finishing. Try replacing ***name*** by your login name or the name of someone else logged in. If you want a list of all users currently logged in, replace ***name*** by an empty string (**""**). Also, change **localhost** to another machine name in your local network; doing so allows you to watch who is logged in on machines at remote locations.

The Coke Machine

Okay, this is not really an exciting application. The result you get is identical to the one you get by typing **finger *name*@localhost** at the shell prompt. So, let's try a really useful application. Today, many Coke machines are connected to the Internet. A short list of such machines can be found at http://www5.biostr.washington.edu/~jsp/coke.html. There, you see that the way to access them is identical to what we did in our first (and not so exciting) example—a finger request. Let us take the first Coke machine from the list and ask the machine which kinds of soft drinks are available there.

```
BEGIN {
 NetService = "/inet/tcp/0/cs.wisc.edu/finger"
 print "coke" |& NetService
 while ((NetService |& getline) > 0)
   print $0
 close(NetService)
}
```

Usually you get a reply with information on the different flavours of Coke and root beer currently available. If you have an account there, you can also order a drink. Many other machines of this kind are connected to the Internet. (See Resources.)

Both examples shown would work even if we deleted the final **close** command, because the operating system closes any open connection by default when a script reaches the end of execution. In order to avoid portability problems, we always close connections explicitly.

The Weather in Germany

Unlike the Coke machine service, most web services we access usually transmit HTML pages across the Internet with a protocol named HTTP. To most people, this is the *real* Internet. Can we access the *real* Internet with GNU AWK? Certainly. We just have to make sure we connect to port 80 of the web server instead of the finger port. This way, we can connect to the Yahoo machine and let it tell us the weather conditions at the place we live.

```
BEGIN {
 NetService = "/inet/tcp/0/
 print "GET http://weather.yahoo.com/forecast/Bremen_DL_c.html" |&
   NetService
 while ((NetService |& getline) > 0)
   print $0
 close(NetService)
}
```

Before starting this script, make sure you know which proxy server your provider uses and insert its name into the second line. If you do not use a proxy, insert the name of the web server (weather.yahoo.com). The result is the HTML content of the web page. It is up to your scripts to bring it into a more readable form or to extract the details of interest for further processing.

Reading the Ticker

Sometimes we are not really interested in viewing a web page. Imagine a web robot (or agent) that looks at the quotes of the Motorola stock shares every 15 minutes and sends you an e-mail if the price hits a certain limit. A command-line call that is executed every 15 minutes is easily written and stored in a shell script. Also, depending on the content of a data file, sending an e-mail is as straightforward to write as a shell script. Here is a script that reads the ticker for you:

```
BEGIN {
 NetService = "/inet/tcp/0/
 print "GET http://quote.yahoo.com/q?s=MOT&d=v1" |&
   NetService
 while ((NetService |& getline) > 0)
   print $0
 close(NetService)
}
```

Again, you must insert your proxy's name into the second line. During execution of the script, a request is sent to Yahoo's quote server and the resulting web page should be redirected to a file by you. With a **grep** command, the price can be extracted from the HTML text and compared to the limit.

Advanced Applications

In these examples, we have seen how useful applications can be written built on the same simple framework. This framework represents only a small

fraction of what can be done with GNU AWK's networking device. Both TCP and UDP connections are available and both clients and servers can be written. More of the advanced applications can be seen in the small manual that supplements the official documentation distributed with the GNU AWK sources. (See Resources.)

Treating network connections like files is not a feature unique to GNU AWK. When TCP/IP was integrated into BSD UNIX in the early 80s, the creators of the socket API originally intended networking connections to appear as special files even to the user. But networking turned out to have many special cases which could not be handled in a uniform way with file handling. Later, the *Portal File System* approach was integrated into BSD UNIX. Portals are similar to GNU AWK's special file but are integrated as a file system into the operating system. This works well because the user can even establish connections at the shell prompt. The most recent implementation of the Korn shell (*ksh93*) provides virtually the same concept (/dev/tcp) at the shell level. None of these approaches has gained wide acceptance among users. Even Richard Stevens' article on Portals (see Resources) has not changed this.

One other approach to networking at the shell level that has gained some acceptance during the past year is the tool **netcat**. Originally a kind of UNIX hacker tool, it simply binds the standard input and output of a process to a network connection. It knows TCP as well as UDP, can behave as a server and allows “port scanning”, i.e., checking if there are servers listening at certain ports. This tool is simple to use and powerful, but some of the comments in the source code are quite unprofessional. Seldom have I seen such a large number of indecent curses, foolish hype and pure ignorance in a source file. Recently, *netcat* has been ported to Windows NT. To a humble user of NT, such a tool is like a long-awaited revelation.

Microsoft Windows

Back to pure *AWKism* and the different forms this belief takes on. On which platforms other than Linux is the networking feature of GNU AWK 3.1 available? It should work on all UNIX systems that comply with the XPG4 rules; this includes every UNIX that has a significant market share. Although the exact release date for GNU AWK 3.1 has not been set, this new feature should also work on Microsoft Windows 95 and NT as a part of the Cygwin tool set as soon as both are out. Cygwin is a UNIX-compatible programming environment that runs on top of Microsoft's Win32 API. It is currently available only as a beta release, but is already able to compile its own set of sources.

When this article was written, compilation of GNU AWK 3.0.3 worked fine, but 3.1 caused problems. If you intend to compile the sources in this environment,

be prepared to experience some trouble. Most importantly, avoid compiling on the same machine you are using for networking with GNU AWK. In case you have only one machine available, reboot between compiling and testing. As of release B20 of the Cygwin tool set, clients and servers written in GNU AWK worked on Windows 95 but no server worked on NT 4.0 SP3. As of release B20.2, the compiler supports linking the file gawk.exe statically with all needed dynamic libraries. This would allow for distributing the GNU AWK interpreter as one single executable, but this executable does not work. Those problems should be solved by the time you read this; therefore, networking on Windows 95 should work.

Trade-Offs

We have seen that network access through a special device is good enough for many useful applications, but there are advanced features we have to trade off for this convenient access method. Some things are simply not possible within the easy-to-use framework AWK employs, namely:

- broadcasting
- non-blocking read
- timeout
- forking server processes

In spite of the lack of these advanced features, advanced applications such as a prototype “web server” or a “mobile agent” have been implemented in GNU AWK. If you need, and can handle, features like broadcasting or non-blocking read, you should use Perl or C instead of AWK.

Resources

Juergen Kahrs (juergen.kahrs@t-online.de) has used AWK on MS-DOS for time series analysis, statistical analysis and graphical presentation, mostly of neurological data. In 1996 he switched to Linux and enjoyed seeing his old scripts still running in a more efficient programming environment. He has come to peace with the fact that AWK will never be mainstream and enjoys seeing C programmers spend nights chasing NULL-pointer accesses. Juergen did the initial work for integrating TCP/IP support into gawk.

Archive Index Issue Table of Contents

Advanced search

Linux Training

Scott Schad

Issue #60, April 1999

A report on Caldera's new Linux Administration Course.

Imagine you are at a piano recital. Expecting the choppy, hesitant performance of a scared youth, you suddenly learn that Van Cliburn is in town and has volunteered to step in so the poor boy can study for finals. What follows is the smooth, effortless performance of a master, an incredible bargain for the price of your tickets. So it is with Caldera's new Linux Administration class.

As Linux completes the transition from religious movement to corporate initiative, the need for serious, accredited training grows. The class composition in this, the second-ever Linux Administration course, hailed from all across the U.S. and reflected the broad appeal of the operating system. Joining me in the pilgrimage to Caldera's Utah offices were an Internet service provider, an IT administrator from a community college, a couple of CAD/imaging specialists, a medical records administrator, an officer from a medical supplies company, a VAR and a system administrator from an on-line catalog firm. Previous Linux experience ranged from extensive to nil.

I came because the Engineering Tools group I manage at MCIWorldcom develops web-enabled databases. I'd been using Linux off and on for several years, but recent testing of the new Linux releases of Sybase and Oracle had convinced me that a significant corporate hurdle to using Linux, the availability of major-league databases, was now gone. Could Linux replace our present NT-based web and database servers? I attempted to find out.

The Linux Administration class runs four days, with an installation class on day five. The course makes few assumptions: you begin by learning how to properly start and shut down the system, configure the boot loader, create partitions and file systems. From there you cover basic system commands, then user and group management. After learning how to customize the environment and set permissions, you move on to a very thorough treatment of networking. We set

up a simple network, configured serial terminals, used NFS and Samba. We synchronized each machine to a time server, configured FTP and web servers and then set up full domain name services.

CGI programming, Sendmail, security and firewalls were covered briefly but thoroughly. After a session on printing and package management, we delved into the arcana of system initialization processes, loading and unloading modules on the fly, backing up the kernel, even rebuilding the kernel to customize it. After a discussion of log files and troubleshooting techniques, we spent the last day installing OpenLinux systems on our class PCs.

Throughout the course, I was impressed with the depth of knowledge the instructor (Wilson Mattos) displayed and the resources Caldera brought to bear on difficult questions. Eric Ratliff (working with Linux since release 0.11) and Allan Smart (a Caldera director) were always on hand to explore involved topics as deeply as we wished.

How does the class compare to other industry training courses? I've taken most of the classes Oracle offers and a good number from Sun. Unlike other industry courses, our instructors at Caldera stayed in the room during exercises to help the students. The instructors didn't read PowerPoint slides to us, and the exercises in the book were relevant, clear and useful. Even given the beta nature of the class and the occasional bobble in the documentation, I would rate the level of instruction and the professionalism as good or better than most industry short courses.

Also to Caldera's credit, the Linux instruction was as vendor-neutral as possible. The class used networked Caldera OpenLinux PCs, but Mr. Mattos always covered variations which might arise with distributions from Red Hat, Debian, Slackware, SuSE, etc. Class participation was invaluable; members contributed many useful tips about distribution idiosyncrasies and about Linux in general.

The little things you'd expect from a Linux vendor were also present—the “Designed for Windows95” sticker strategically placed on the copier waste basket, abundant refreshments, the vigorous debate that arose over future Linux certification tests. (Does complying with the GNU open-source license mean Caldera must include answers with the certification questions?)

The one-day installation class at the end of the week clarified the often obscure aspects of loading up Linux. In addition to giving everyone a boxed copy of their latest Linux release, Caldera instructors even helped class members get Linux up and running on their personal laptops.

I would have preferred more elementary coverage of certain basics (shell scripting, for example) at the expense of advanced networking instruction, but the class did a good job of distilling the core knowledge of Linux into a retainable, five-day package.

Will I now go back and lobby MCIWorldcom to put Linux on every desktop? Probably not. Even with the slick new KDE X interface in Caldera's latest distribution, I think it is still going to take further evolution before the corporate desktop line is breached. However, my company places a premium on finding the most cost-effective, efficient ways to run its telcom business, and ignoring opportunities has never been a winning strategy. I think Linux can find a secure home right now on machines used as corporate servers. No matter how you slice it, running Linux on network file, print, mail, database and web servers delivers potent bang for the buck. These systems scale quite well, are remarkably stable and are least restrictive in terms of development options. Major league databases are now running quite well on Linux (Sybase comes bundled with the latest OpenLinux release) and the remaining corporate concerns about support and formal training are being handled capably by companies like Caldera.

I'd like to thank the Caldera staff for good value on the education dollar and for their hospitality during the class. I'd also like to thank the Engineering Tools group at MCIWorldcom for tolerating my Linux evangelism.

Scott Schad is a petroleum geologist (when the price of oil is high) or a database developer (when it is not) in Tulsa, Oklahoma. When not pushing Linux and True Basic, he studies Bruce Lee's Jeet Kune Do to become a lean, mean, fighting machine.

Archive Index Issue Table of Contents

Advanced search

Blender

Ben Crowder

Issue #60, April 1999

No, it is not that thing you use to stir up food in your kitchen—it is a hot new state-of-the-art 3-D modeler.

Blender is a free, full-featured 3-D modelling and rendering environment. In the words of the Blender web site:

> Being the in-house software of a high quality animation studio, Blender has proven to be an extremely fast and versatile design instrument. The software has a personal touch, offering a unique approach to the world of three dimensions. Use Blender to create TV commercials, to make technical visualizations, business graphics, to do some morphing or to design user interfaces. You can easily build and manage complex environments. The renderer is versatile and extremely fast. All basic animation principles (curves and keys) are well-implemented.

If you have any previous experience with 3-D modelers, one of the first things you will notice after playing around a bit with Blender is that it is incredibly fast. The primary author, Ton Roosendaal, says one of his favorite hobbies is throwing extraneous code out of Blender, and it shows.

Blender History

Way back in 1989, Ton Roosendaal started up an animation studio called NeoGeo with a handful of friends. They wrote their own in-house 3-D software, “Traces”, but Ton was not satisfied with it. He evaluated it (as well as SoftImage and Alias), analyzed its shortcomings, and in 1995 started work on Blender. In January 1998, NeoGeo released the SGI version of Blender. Due to the immense popularity of the program, Linux and FreeBSD versions were soon released in April. In mid-1998, NeoGeo was partly taken over by another company (Alatis),

so Ton created a new independent company, “Not a Number”, to take over Blender development.

Getting Blender

The main Blender web site is at http://www.blender.nl/blender.html. The software is available at NeoGeo's FTP site, but it is best to find an FTP site geographically closer to you. Linux and FreeBSD x86 versions are available, but if you have an Alpha, SPARC or Power PC, you are out of luck. Hopefully, ports will soon be released for other platforms. However, there is a broader availability for SGI's—from R3000-based Indigos all the way to the R10000-based Octanes.

Download the archive file you want, either blenderLinux-dynamic_*x.xx*.tar.gz or blenderLinux-static_*x.xx*.tar.gz, where *x.xx* is the version number. (The current version at the time of this writing is 1.37.) The dynamic version is linked to the Mesa 3.0 libraries as well as a few others, so if you don't have Mesa, get the static version.

If you want, you can also grab the Blender tutorial files (tutor_1.01.tar.gz and tutor_2.0.tar.gz). The version 1.01 tutorials aren't actually tutorials—they are example files, but they don't show you how to create the files. Regardless, try them out, since they are pretty cool. Examples include a bat animation, a short sequence from “The Lost Ride”, a 3-D game by NeoGeo and a cool walking spider animation, in addition to a bunch of others. The version 2.0 tutorial shows how to morph a cube into a sphere and includes a tutorial on how to do the morphing. In addition to the two tutorial files, you can get the Blacksmith demo which includes examples of particle animation (i.e., fire and smoke).

I would suggest getting the manual—manual_1.02.html.tar.gz. While it is a little out of date, some of the information still holds true. It especially helps in understanding how Blender is object-oriented.

System Requirements

Even though the Blender web site says the minimum system is a Pentium 90 with 32MB of RAM, I've found that a fast 486 (DX2 66 or better) with 16MB of RAM is adequate for simple modelling and even for some more complex stuff. Still, if you want to do serious modelling, you definitely need at least a P90. As noted above, however, at this time you must have an x86 machine. You will need at least an 800x600 resolution screen (the buttons are a bit squashed though, so if you can get 1024x768 or better, go for it) and you definitely need at least 15bpp. You might be able to get away with 8bpp, but I would heavily advise against it. For one thing, everything is green, and in some rare cases, Blender might even crash.

Installing Blender

1. Once you've downloaded the Blender archive file, move it to the installation directory and unpack it:
   ```
   gunzip -c blenderLinux-xxx.tar.gz | tar -xv
   ```
2. Change directories to the blenderLinux_xxx directory (**cd blenderLinux_xxx**).
3. Copy all of the .B files to your home directory (**cp .B*h ~**).
4. For bash users: edit your /etc/profile file (or if you're installing locally, $HOME/.bash_profile) by adding a line that reads:
   ```
   export BLENDERDIR=/
   ```
5. For tcsh and csh users: create an environment variable **BLENDERDIR** in your $HOME/.cshrc file:
   ```
   setenv BLENDERDIR /
   ```
6. Restart your shell.
7. Run Blender (**blender**).

Basic Usage

If you haven't used 3-D software before, the GUI may look complex and difficult at first, but you'll get used to it. When you first start up Blender, you'll see a large grid in the center of the screen—that's the 3-D window. Down on the bottom is the buttons area. Finally, at the top is the options area, where you choose which scene to edit, font paths, etc. If you are running on a small display (800x600 or less), you may not be able to see most of the top window. In this case, click on the edge of the top window and drag it down until you can see it. For now, you need only be concerned with the 3-D window and the buttons window.

Basic keystrokes/mouse movements

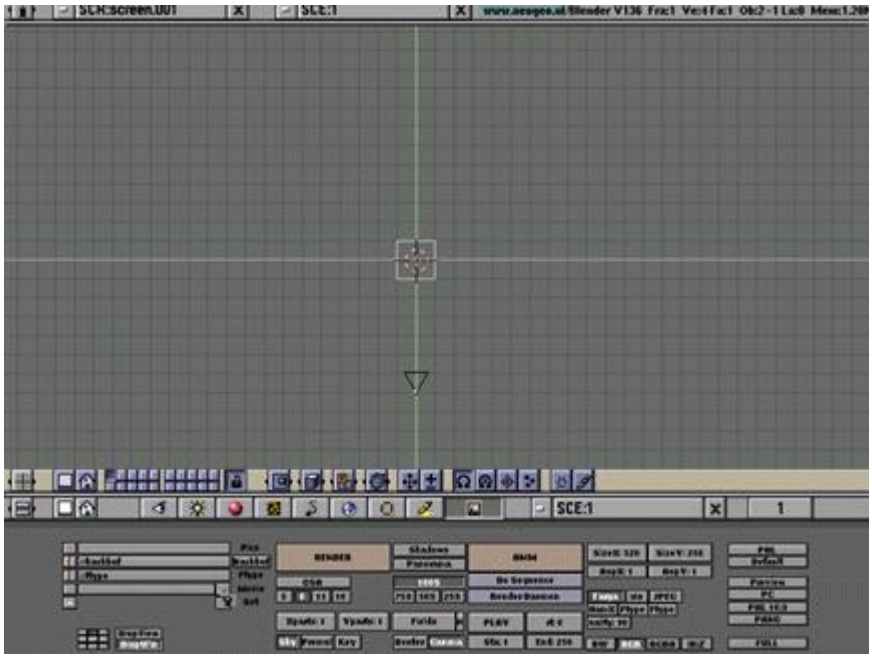


Figure 1. Initial Blender Screen

All of the Blender widgets and menus are rendered through Mesa; this means you can zoom in and pan around almost anything. For example, in the buttons window (down at the bottom), you can hold **Ctrl-MiddleMouseButton** and drag around to zoom in or out. Notice that when you zoom, the buttons, including the fonts, are scaled. To pan, use the middle mouse button and drag around. This works everywhere—even in the button bars on each window.

The black cross with a red and white striped circle around the center is called the 3-D cursor. Whenever you add an object, it will be placed at the location of the 3-D cursor. To move the 3-D cursor, just left-click on the spot where you want it.

The black triangle with the little yellow ball (a few units down from the center) is the camera. Right-click on it to select, then press **g** to move it around. When you decide on a location for the camera, click the left mouse button.

The Blender windowing system is somewhat similar to HTML frames. Each window can be infinitely split (though a practical limit does exist—if you can't see the windows any more, you've gone too far) by moving the mouse into the window to be split, then clicking with the middle mouse button on the window pane perpendicular to the direction of split. If you want to split a window horizontally, click on either the left or right vertical pane. Then when “Split?” comes up, click on it. To join two split windows, right-click on the pane you want to remove, then click on “Join?” when it comes up.

Two main editing modes are included: normal and edit. You can toggle between the two using the **tab** key. When you add an object, Blender automatically switches to edit mode (for most objects, anyway). For example, let's say we add

a Mesh cube. When the cube appears on the screen, it is in edit mode. If you right-click on one of its vertices (the purple dot means it is not selected, the yellow dot means it is), you can then press **g** to move the vertex around. If you press **b**, which stands for "Border Select", you can draw a rectangle over the selected vertices.

The Toolbox is brought up using the space bar and enables you to add objects. Note that you can also use the **Add primitive** function (**shift-A**), which brings up the Toolbox and the Add submenu. Most of the hot keys are placed here. If you need to get out of the Toolbox window, either press **esc** (the standard for Blender windows) or move the mouse away from the window.

Figure 2. Building a Blender Scene

Your First Scene

To build your first scene, follow these steps:

1. Switch over to top view, if you aren't already there (number pad 7).
2. Move the 3-D cursor to the center of the grid.
3. Bring up the Toolbox (space bar).
4. Left-click on Add, then on Mesh, and finally on Cube. Another way of saying that is **Add->Mesh->Cube**.
5. Press **tab** to leave edit mode.
6. Click on the red sphere in the button area (Materials).
7. On the right side of the screen, click on the icon with a white horizontal bar.

8. In the left side of the buttons area are some sliders labeled R, G and B (Red, Green and Blue). Slide the Blue all the way to the right, and slide Red and Green all the way to the left. The “material preview” rectangle should turn blue.
9. Move your mouse cursor back into the 3-D window.
10. Change to side view (number pad 3). Move the 3-D cursor to somewhere above the cube, though it should be somewhat close (no further than 20 units away).
11. Click on **Add** then **Light**.
12. Press **f12** and watch the scene render.
13. Press **f11** to get rid of the render window.
14. Press **f2** to save the created scene. Once the file window comes up, click in the second input box from the top (under the directory name) and enter a name for your scene. Then press **enter** twice to save the file.

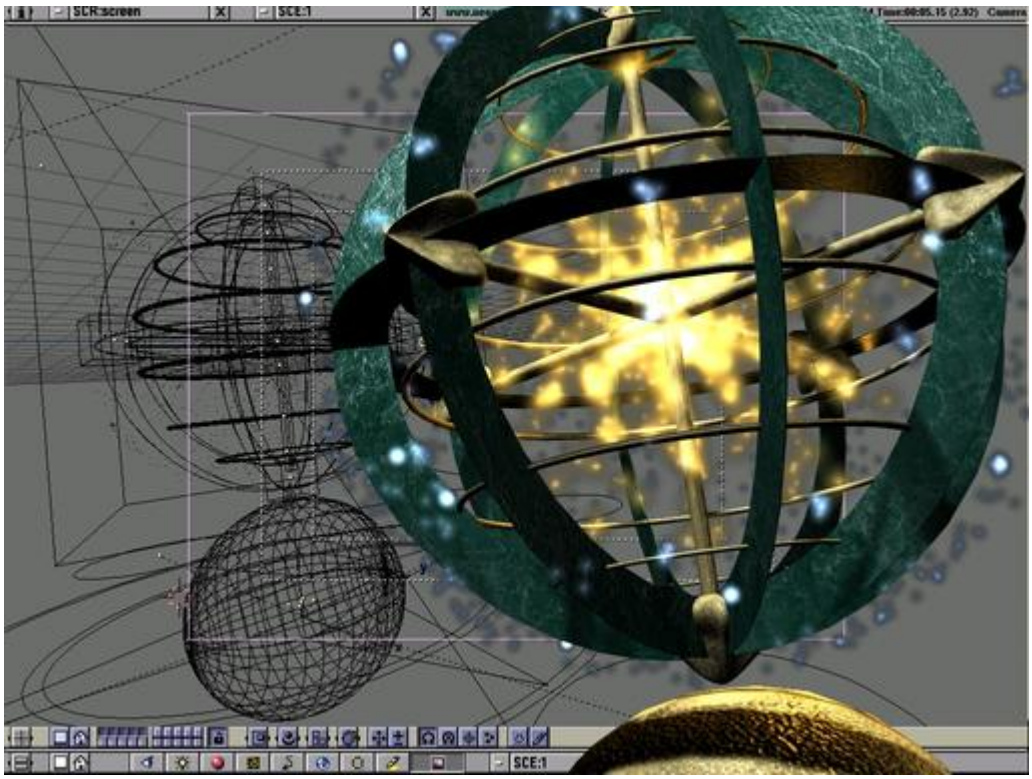


Figure 3. Blender Designed Scene by Bart Veldhuizen

The Future of Blender

Ton Roosendaal hopes to eventually port Blender to a number of systems, but right now that isn't a priority. By the time this article is printed, the manual and a free version of Blender 1.5 will have been released. After the manual goes out, a commercial CD-ROM will be published containing Complete Blender (version 1.8)--hopefully, by mid-1999.

Resources

Ben Crowder is a young Linux aficionado living in Utah. He finds himself becoming more and more hopelessly addicted to Linux. In addition to fiddling with the insides of computers, Ben enjoys reading, writing and music. He can be reached at mlcrowd@enol.com.

Archive Index Issue Table of Contents

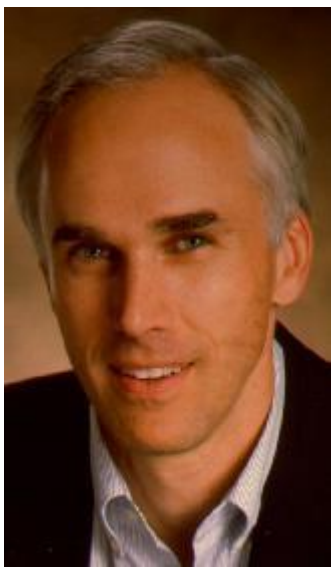
Advanced search

LJ Interviews John Ousterhout

Marjorie Richardson

Issue #60, April 1999

LJ talks to the creator of Tcl/Tk about the port of TclPro to Linux.



John Ousterhout provided the Open Source community with the freely available Tcl scripting language and its companion toolkit Tk. These days he is busy running his company Scriptics, where he is developing commercial tools for use in developing Tcl/Tk applications. I talked to him by phone in December 1998, when Scriptics released TclPro 1.1 for Linux.

Marjorie: The first thing I would like to know is what is the big deal with porting to Linux? Haven't Tcl products always worked with Linux?

John: The big deal! You have to distinguish between Tcl the language and the TclPro product. Tcl the scripting language, its toolkit Tk and all the various extensions available for them have worked on Linux for years, probably five years or more. They are present on most of the popular Linux distributions, various CDs by Red Hat and others. The news we are talking about right now is our company's TclPro product, which is a set of development tools. We have

just made a new release that supports Linux for the first time. Our previous releases did not support Linux, although you could run Tcl programs on Linux, our development tools which make it easier to develop Tcl scripts were not yet available to you.

Marjorie: What operating system were they running on?

John: Scriptics is a fairly young company—we've been around just about a year now. We released our first product, TclPro 1.0, in September and it ran on Windows 95 and 98, NT and two UNIX variants, Solaris and HP-UX. We thought about including Linux support from the start, but we guessed that since Linux was a free platform that not many people would be interested in buying products for it. The TclPro Toolset is a commercial product.

So, we didn't initially support Linux. We thought we'd probably do it eventually, just not right away. Then as soon as we made the first beta release last summer, we were inundated with requests for Linux—it was truly a pleasant surprise to see how much interest there was from the Linux community. It began to appear to us that, in fact, Linux people are actually interested in this kind of tool and would even be willing to pay for it.

Marjorie: It's amazing, isn't it?—that they'd be willing to pay!

John: Well, my take on this is that Linux may have started off with a community of gung-ho users who were attracted by the freeness of it. I think more and more people are now using Linux because of the sophisticated features and reliability it provides, and the fact that it is free is a bonus. People are willing to pay for software that adds value to their Linux systems.

Marjorie: I think they are too, definitely.

John: Once we started getting all these requests for Linux, we realized we had goofed in not planning for Linux in our first release. We looked at getting it into that 1.0 release, but it would have messed up the schedule too much. So, we planned a 1.1 release as soon as possible after the first release. That's what we are doing right now. It's been about three months since we made the 1.0 release. The 1.1 release adds a port for Linux and the SGI platform.

Marjorie: How much work did it take to do the port? We've heard from some people that all they've had to do is run **make**.

John: Yeah, Tcl is very portable, and so was TclPro. The only hassle was actually getting a Linux machine here and getting the build environment set up with C compilers, etc. Once we had that done, it was only a day or two of work to get TclPro running on Linux.

Marjorie: We like to hear that it was quick and easy.

John: One of the neat things about Tcl is that it is an extremely portable programming environment—Tcl, all its applications, and all the TclPro tools are actually built with Tcl. Thus, they are all very portable. In some sense, this is a relatively minor announcement for Scriptics in that it is an upgrade release. It has a couple of new features but nothing earthshaking in the way of features. However, I think it's a very interesting release as yet another confirmation of how much demand there is for commercial products for Linux. I also think it is an interesting confirmation of how important Tcl is to the Linux community—people who use Linux are crying out for it. Tcl is one of the best ways of doing graphical user interfaces and also a variety of other applications on Linux.

Marjorie: I think that's true, too. What advantages do you think Tcl has over other scripting languages?

John: That's a good one. The real power Tcl brings is that it's wonderful for creating integration. Compared to other scripting languages, Tcl's advantage is that it can be used for a huge variety of applications. Most other scripting languages were designed for a particular domain. For example, Perl was initially designed for system administration and reporting tasks; JavaScript was designed for scripting in web browsers. These languages work great in the areas for which they were designed, but sometimes don't work as well in other areas (for example, JavaScript isn't used much at all outside web browsers). Tcl was designed to be used in many different areas, and it has worked well that way. Thus, if you have a variety of things to integrate, Tcl is likely to provide the most flexible platform.

Marjorie: Which tools included in Tcl do you think will be most appreciated by Linux developers?

John: Well, definitely the debugger—all programmers need that tool. Also, the Error Checker and the Wrapper will be useful for Linux programmers. Actually, after thinking about this some more, I think all four of the TclPro tools will be useful for Linux developers. Linux developers are not that different from other developers, so I think they will use all the same tools.

Marjorie: Tcl has always been Open Source; TclPro is commercial. How does Scriptics intend to balance this act in the future?

John: The core code for Tcl and Tk will always be freely available. This is a commitment we have made to the Open Source community. What we will do commercially is develop products such as TclPro that aid in Tcl program

development. If we gave all our products away, we wouldn't stay in business long.

We allow people to download TclPro from our web site with a free 30-day evaluation license. Since the Linux version became available in beta form at the beginning of November, it has been the number two platform in terms of popularity. About 55% of the evaluation downloads are for Windows (95 and NT). Linux is second with about 21% of the downloads. Solaris, which we had expected to be the number two platform, is a distant third with only 13% of the downloads.

Marjorie: Do you think the time will come when free software will do away with commercial software altogether?

John: No, but I would be happy to be proven wrong.

Marjorie: Some say that with free software, programmers would make their living by consulting and support.

John: That's an interesting idea, if you think about the personality of someone doing this. They're most likely going to build something they want to use themselves, then give it away to others. They tend not to produce other types of software, leaving many areas open for commercial software development.

I think that's why you see operating system cores, scripting languages, mail delivery tools, web servers and other things which expert programmers enjoy using themselves as open source and you don't see enterprise resource planning applications or even things like word processors and spreadsheets yet.

Marjorie: Well, there are some applications out there—they are just not very far along.

John: People like Eric Raymond think we are just at the beginning of the evolutionary development of open source software, and that all of these other applications will come about in time. Maybe he's right. My personal hunch is that open source development is mostly going to be the platform and tool kind of software.

Marjorie: I think the two examples Eric gives for spreadsheets and word processors are Maxwell and xxl.

John: Well, I'd be happy to be proven wrong on that too, you know.

Marjorie: Well, I've always wondered how it would work. As you say, people somewhere along the line have to make a living.

John: I think the whole open source movement is incredibly intriguing and an amazing and empowering device for the individual. One smart person with a good idea and a bit of spare time can harness the energy of thousands of other people and build software that gets used by hundreds, thousands or millions of people. How many industries in the world can be found where such an incredible impact can be made by a single person? It's a really neat thing that smart people with good ideas can change the world.

Marjorie: Someone told me recently that they heard you speak at one of the conferences and that you didn't feel the open source movement currently addressed the needs of the unsophisticated or new user. Is that true?

John: I think it is often true. Again, it gets back to the mindset of the open source developer—that they are typically building programs for themselves. So when they make programs available to others, they tend to have a level of documentation and installation appropriate for a fairly sophisticated programmer rather than for a novice programmer. Again, this is an area where commercial efforts can fill in the gaps by providing the means to make things even easier to use. A great example of that in the Linux sphere is the arrival of companies like Red Hat, Caldera and SuSE which have provided Linux distributions that are much easier to install than they used to be.

If you look back about two or three years before those companies existed, it was a much bigger proposition to install Linux—not for the faint of heart!

Marjorie: What motivated you to decide to write a new scripting language in the first place?

John: Like many other people doing open source stuff, I was solving a problem of my own. The problem I had was that I was building a whole bunch of different tools that needed to have command languages in them and I wanted them to have a powerful command language. I didn't have the time to build a separate one for each tool. So the idea for Tcl was to build a small interpreter that I could plug into different tools, thus using the same basic language as the command language for many different programs. I did that by building an extensible language; Tcl provides all the basics. It can then be dropped into an application with all of the application's features hooked into Tcl as extensions adding to the basic features of the language. That way, I have a very powerful command language in every program, and all programs share the same core part of the language—the Tcl interpreter. That was the basic idea. From there, I added an extension to do graphical user interfaces called Tk. Then we suddenly

discovered you can actually write many interesting Tcl scripts without putting Tcl inside another application, but instead using it stand alone. With Tcl and Tk together, you can write great GUIs and other sorts of integration applications. Once that happened, usage by other programmers just exploded.

Marjorie: One of the first articles we got about Tcl/Tk was about using it to write a GUI.

John: That ability wasn't the first thing on my mind when I was writing Tcl, but it turned out that in solving my problem, I had also built something that was very easy to use for developing general purpose GUIs.

Marjorie: How about some personal type stuff about you? What do you do for fun?

John: I do things with my family for fun. Between being CEO of a startup and trying to be a good father to my kids, there isn't a lot of time for anything else. But I enjoy lots of activities with my family.

Marjorie: Okay, how about where did you go to school, that sort of thing?

John: I was an undergraduate at Yale University in the early 70s and got my Ph.D. in Computer Science from Carnegie Mellon University in 1980. I was a professor at Berkeley from 1980 to 1994, when I went to Sun. I ran a Tcl development team at Sun until spinning out to start Scriptics in January of 1998.

Marjorie: Thanks very much for your time.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Linux Certification for the Software Professional

Tobin P. Maginnis

Issue #60, April 1999

A discussion of the need for certification and a proposal from Sair, Inc. for a Linux certificate program.

Within the Linux community is a growing discussion concerning the need for certification. Although certification may not be appropriate for today's Linux enthusiasts, it will be essential in the future as Linux software is brought into corporate and government environments. Certification programs and open source software will become significant as more and more independent professional programmers integrate Linux and Linux-based software into standard contract programming. Use of the certification process to develop minimal standards that are acceptable to the Linux community will also smooth the coming transition to state board licensure of the Professional Software Engineer.

When comparing diverse professions such as law, civil engineering, accountancy, clinical medicine, clinical psychology, and even hairdressing, one notices they all have a requirement of certification or licensure for the practicing professional. Computer professionals, on the other hand, do not have such a requirement.

The main reason for the lack of required credentials is probably that the computer discipline is only about twenty-five years old and, therefore, it has yet to come of age in industry or academia. As evidence of this field's youthfulness, traditional engineers often dismiss computer science as pseudo-engineering. Programmers stress the art (as opposed to science) of programming and the art of system administration. Computer science degree programs have been placed in university liberal arts, business or engineering schools. Meanwhile, computer scientists themselves cannot agree on even a short list of essential degree requirements for the Bachelor of Science in Computer Science (BSCS).

In spite of this awkwardness and uncertainty, our profession is doing well. BSCS graduates command high initial salaries that match or exceed other engineering disciplines. Computers control much of the country's infrastructure and this control is dramatically demonstrated by the manpower and resources currently employed to address the year 2000 problem.

In other words, the debate over certification in general and Linux certification in particular represents the growing pains of a relatively new profession and society's need to understand what a software engineer does. To make matters worse, the U.S. Department of Labor predicts that the computer industry will have 300,000 more programmer positions than programmers over the next few years and universities will not be able to fill the predicted demand. In fact, overall enrollment in computer science programs has decreased slightly in the last year (see http://stats.bls.gov/oco/ocos110.htm).

The role of certification, therefore, is to provide a method by which employers can specify and expect a given level of computer system expertise from employees without requiring them to have an advanced computer science degree. In a secondary role, certification acts as a supplement to a college or university degree by providing professional development through continuing education.

It is now time for the Linux community to shape the future of the software engineering profession by agreeing upon minimal industry requirements that will be codified into certification and licensure requirements.

Why a Linux Certificate?

A Linux system and its array of development tools and applications represent a central core (or “common body”) of knowledge for today's professional software engineer. Linux “empowers” the software professional with a set of tools that allows one individual to quickly provide sophisticated, flexible and reliable system solutions on a variety of computing platforms. These sophisticated solutions would otherwise require teams of programmers working months to integrate proprietary and incompatible software modules.

Second, a fundamental concern of corporate and government managers is the cost of operations—and Linux dramatically reduces this cost. The reduction is not so much from the idea of free software, but more a result of increased programmer productivity through complete control over the system. Another aspect of reduced cost is that the entire Linux community, including business and government, is continually debugging and improving the program base, making the system even more cost effective with each new distribution.

A third reason for a Linux certificate comes from state licensing boards. Licensing is a governmental action that seeks to safeguard the health, safety, welfare and property of its citizens and businesses through regulation of the offer of services, review of performance and use of the title “Engineer”.

On February 18, 1998, the Texas Board of Professional Engineers stated its intention to recognize software engineering as a legitimate sub-discipline. On June 17, 1998, the Texas Board approved a license for the Professional Software Engineer, and on August 3, 1998, began licensing software engineers through a waiver process (http://www.main.org/peboard/).

Given how Linux is used, this is a significant move by the Texas Board of Professional Engineers. Linux system software allows the independent software professional to provide a competitive bid, program the application and deliver a software product quickly. The professional programmer knows that Linux is a complete and open source software system that will simplify the solution of any development problem. In other words, from embedded systems through advanced networked clusters, Linux has already handled many programming and integration issues. The software developer is free to configure Linux in any way that best suits the required solution. Hence, Linux is an integral part of the software engineering process and as State Board licensure continues to grow, Linux must be a part of the licensing process.

Finally, given the predicted demand for software professionals, a Linux certificate will provide a valuable aid to employers in discovering productive software professionals.

The Certification Process

Certification is an assessment of an individual's knowledge and skills in a given area of expertise. Currently, more than 170 certificates are offered and many of these have overlapping areas of expertise. These certificates are generally centered around a vendor's hardware or software product.

The certificate provider typically defines core knowledge and develops texts, study guides and other course material that cover the defined expertise. Students enroll in the certificate program and purchase the appropriate texts and study guides. Supporting materials such as video and/or audio tape lectures are sometimes offered for an additional fee.

Core knowledge is usually divided into four to six areas, with separate tests for each area. The goal of such test taking is not to compare one person's test performance to another's, but to determine whether an individual meets or exceeds the minimal requirements of the vendor training program. A typical

student (given family and job obligations) takes the tests over a six- to twelve-month period.

Once the candidate earns a passing grade on all of the required tests, a certificate is issued authorizing the software professional to list the certificate with their name. The certificate provider usually offers “perks” that include a technical hot line or exclusive forum with other certificate holders, a WWW database listing of member skills and services, plaques, photo ID badges and logos for business cards and letterheads. The cost of obtaining these certificates can range from $2,000 to $10,000 US.

Key Certification Benefits

Certificates assure employers that the holder has a specific set of skills and a minimal level of competence. Company management, in turn, exploits certification by advertising the employee's skills to their customers and billing customers more per hour than for non-certified employees.

Certification typically gives an employee more job options and yields better pay. Certification in a high-demand area can lead to a wage increase of $5 to $10 US per hour (that's $10,000 to $20,000 US per year) over non-certified computer professionals.

Many corporate managers consider certification an investment in human resources and, as a result, they pay for or otherwise assist with certification costs. The consensus among managers is that certification leads to improved employee service, reduced problem-solving time, easier assimilation of new technology and higher overall employee productivity.

It Is Time for Linux Certification

Several issues have come together that indicate it is time for us to offer a Linux community-based certificate. First, there is a clear consensus among the Linux community that Linux should continue to grow, receive wide acceptance among programmers and should offer an alternative to traditional software products. However, the growth of Linux functionality and acceptance is not a simple straight line on a graph. Over the last two years and especially within the last year, Linux distributions have reached a “critical mass” of stability, reliability and completeness that will accelerate the growth of Linux over the next few years. Second, the expected rapid growth of Linux will create a need for business and government managers to have an additional metric in the selection of new employees as well as an additional metric for employee merit pay increase and employee promotions. Third, as other State Boards of Professional Engineers see what has been established in the State of Texas, it will be only a matter of time before licensing software engineers is common.

Consequently, a Linux certificate will act as a conduit to build a community consensus on certification and licensing.

Hence, now is the time for a Linux certificate and it is Sair, Inc.'s intention to act as a conduit for the Linux certificate by describing our certification process in detail, soliciting critical analysis, debating the specifics and creating a Board of Advisors. The Board of Advisors will consist of respected Linux community members given the responsibility of ensuring that the certificate meets minimal community standards and covers material representative of how Linux is used in business, government and independent programmer shops. Furthermore, the Board will recommend actions to advance Linux in the computer industry.

A component of this advancement will be combining the accumulated certificate knowledge with activism to ensure that the Linux community is represented at the State Board licensing level.

The Linux Certificate

The proposed Linux certificate will cover four areas: installation, networking, shells and commands, and security. In the future, and depending on the directions Linux takes in business, a Linux kernel certificate that covers the details of kernel development, maintenance and embedded systems will be considered. There will be tests for each of the four areas and candidates can take them one at a time or in clusters. Hands-on demonstrations may be an essential component of some tests.

Examination Topics

A working draft of a comprehensive outline of the proposed certificate material is at http://www.linuxcertification.org/. Although the outline may be missing a given topic or detail, a surprising amount of information must still be covered. Given the large amount of Linux system information that exists, we have yet to define the exact breadth versus depth of the exam. For example, it is assumed in the study guides that the student already understands basic operating system concepts such as memory and process management, interprocess communication, file management, device drivers, system abstractions, network layers and so on. A separate section devoted to these concepts may have to be developed in the future.

Candidates will need additional background study material to support the “narrow” study guides. Currently, we are employing existing Linux texts to provide background and related details for the Linux certificate topics. In the future, we will integrate other on-line documents into the study guide through either cross reference or additional independent study guides. This means that, at least initially, the Linux student will require a larger textbook library than that

required for other certificates. (See the web site for a suggested reading list of background texts.)

Existing study guides (with background text references) are being refined and more are being developed. They can be reached from the “study guide” hyperlinks in the working draft outline, for example at http://www.linuxcertification.org/introqa.html. These guides are in the form of essay questions followed by answers.

Exam Administration

Members of the Linux community would clearly like to take certification exams on the Web. (See http://www.linuxjournal.com/HyperNews/get/certification.html.) Unfortunately, no straightforward method exists for user validation (fraud prevention) over the Web. Assuming the Linux certification process becomes a self-sustaining enterprise, we will be developing fraud-proof technology for web-based exams. Until that time, we will use traditional testing methods and environments. Initial tests will be based upon “book knowledge”, while advanced tests may include hands-on demonstrations. Assuming enough interest, book knowledge tests can be given in commercial testing centers such as Sylvan Prometric. Testing on advanced material and hands-on demonstrations will be made available as the need arises.

Milestones for the Linux Certificate

These items need to be completed:

- Solicit community comments. (Initial outline at http://www.linuxcertification.org/.)
- Complete development of study guides, using selected background texts and on-line material as reference. (Initial study guides are hyperlinked at http://www.linuxcertification.org/.)
- Use community comments and Board recommendations to improve study guides.
- Upon completion of the first study guide, accept candidates for the certificate.
- Based upon interest, schedule tests while study guides for each area are developed.
- Upon completion of all study guides, schedule final tests and begin awarding certificates.
- Begin active consulting with State Boards concerning Linux and its relationship to licensing.
- Develop continuing education credits and courses.

Conclusion

In planning for the future, we are developing a Linux Certificate now. We want the certificate to be truly representative of the Linux community, and your comments are essential in helping us achieve that representation. We believe certification has significant mutual benefits for employers and employees and will become a key tool for managers as Linux software is brought into corporate and government environments.

As more and more independent professional programmers integrate Linux software into their standard contract programming, we will be promoting Linux as core knowledge in the State Board of Professional Engineers licensure process.

Resources

Acknowledgments

Dr. P. Tobin Maginnis (tm@sairinc.com) is an Associate Professor of Computer Science at the University of Mississippi and the President of Sair, Inc. His areas of specialization are operating systems, networking, distributed operating systems and multimedia. See http://www.cs.olemiss.edu/~ptm/ for more information. As President of Sair, Inc., Dr. Maginnis supervises a programming shop that provides custom programmed software solutions to clients in the Chicago area. See http://www.sairinc.com/ for more information.

Archive Index Issue Table of Contents

Advanced search

Arkeia

Charles Curley

Issue #60, April 1999

The product is aimed at professional network administrators, but don't let that scare you off.

- Vendor: Knox Software
- E-mail: sales@knox-software.com
- URL: http://www.knox-software.com/
- Price: Variable, depending on the number of client computers. The shareware price is user-defined.
- Reviewer: Charles Curley

Arkeia software presents a backup solution to administrators of heterogeneous networks of all sizes. From a one-computer shop to a large shop with hundreds of computers, Arkeia is designed to work on any size IP network. It also works cleanly with a mix of widely different operating systems. The product is aimed at professional network administrators, but don't let that scare you off. It is documented well enough that an amateur who is reasonably knowledgeable about Linux and his organization's network can set up and run the software. The product installs easily on Linux and on Windows 95. This is the kind of product that will make Linux a serious contender as a real-world operating system.

The server software can run on Linux, Solaris, SGI, AIX, HP-UX or SCO. Give it a fast connection to your network and one or more fast tape drives, and you are ready. Clients, in addition to the servers just listed, include Windows NT, Windows 95 and 98, Novell, VMS, ICL and others. The servers and clients

communicate over TCP/IP, so the physical layer of your network doesn't matter. You don't need Windows sharing or NFS to do your backups.

To review the software, I downloaded the shareware version. I set up the server on my 333 MHz Pentium II Red Hat 5.0 Linux box and put the Windows 95 client on my 166 MHz Pentium laptop. They are connected over a TCP/IP network on Ethernet. The installation and operation manuals are in Adobe PDF format.

The software for Linux comes in gzipped tar files and contains its own installation software. The installation software inserts the NLSERVD daemon into the boot sequence and starts it running, so as soon as you finish the installation you are ready to use the product. Knox does not offer an RPM version. The software goes into its own /usr subdirectory and uses one file of its own under /etc, so there is no conflict with an RPM system.

The Windows 95 client software is delivered in a professionally done InstallShield package. The average Windows user who can install his own software will be able to install it with no problem. It correctly saves and restores Windows 95 attributes—something backups over SAMBA to a non-Windows tape server won't necessarily do. The Windows 95 client will not back up a file while Word 6 has it open, but will get the automatic backup file if there is one. This is a failing of Windows 95, not Arkeia; the only workaround is for users to shut down programs when they leave for the day.

The interface for the software is a GUI using X on Linux. It was a bit disconcerting when I first saw it but is internally consistent, and I quickly learned to use it. Tool tips make the software much easier to use.

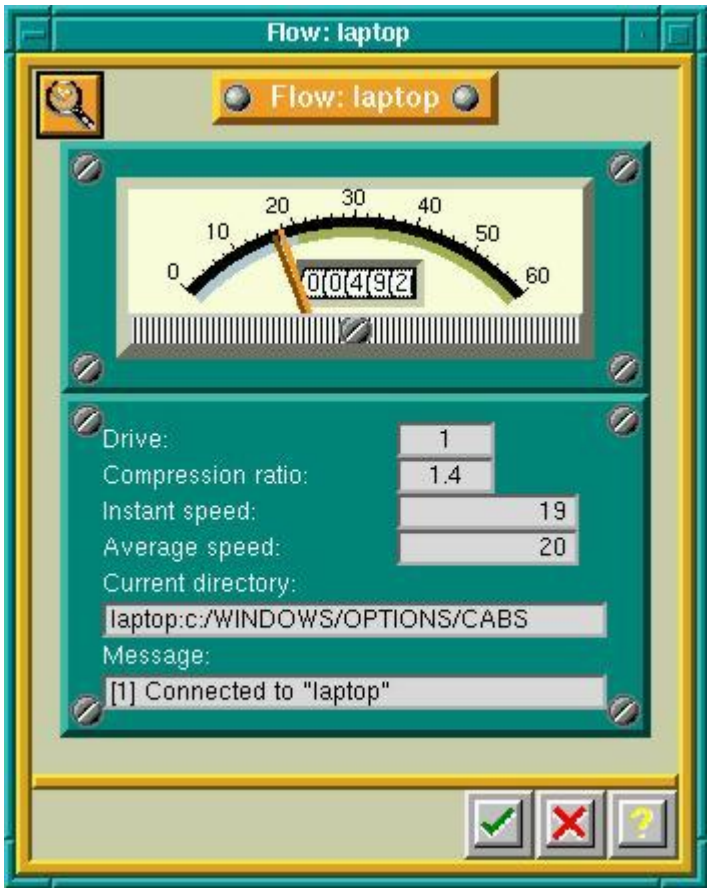


Figure 1. Backing up a Windows 95 laptop showed an average rate of 20MB per minute and a compression ratio of 1.4 to 1.

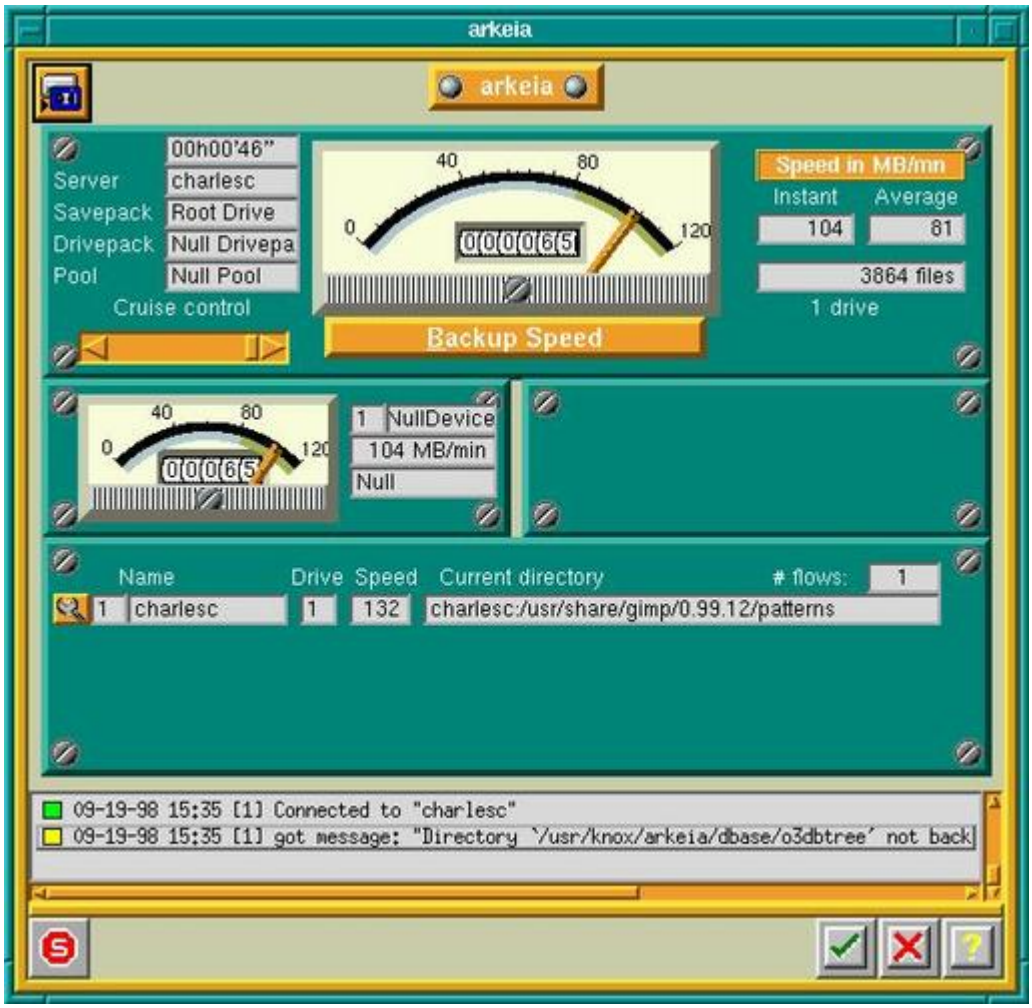


Figure 2. The main backup window shows a test backup to the null device. 65MB into the backup, the average data rate of 81MB per minute shows what the software could do on a Pentium II processor at 333 MHz. Note the error message with the yellow flag, which will be logged.

Because the software is clearly designed to support large shops, Arkeia is conceptually more complicated than typical backup software such as BRU. Before you can even do a test backup, you must define a number of objects.

- You must define tapes to the system. Provisions are made for writing labels with bar codes for use with robot arms. (Since I don't have a robot arm, I was unable to test that part of the system.)
- Tapes are combined into tape pools. The software rotates the tapes within each pool as needed.
- You also define “savepacks”. These are the computers and directories on the computers to be backed up in units. For example, I have a savepack that consists of my laptop. I could easily define a savepack that consists of several computers (say, a department of a company) or one that consists of a single directory on one machine (such as an engineer's CAD drawings).

Customer support, even for the shareware package, is excellent. I made an error and e-mailed the support address for help. They replied with a preliminary diagnosis and a request for some log files. I returned those and got a diagnosis back within 24 hours of my first e-mail. Try that with any large software vendor. I was favorably impressed. I also made some suggestions for improving the product. The return e-mail indicated that my suggestions will be seriously considered. Again, Knox compares favorably with many software vendors.

One thing I like in a backup package, but is missing, is a manual retension command. Of course, I can use **mt retension** on most any UNIX or Linux box. Arkeia also has a provision for a user-defined command-line operation before and after each backup. If you like retensions before each use of a tape, add it here. This is customization I have not seen with other tape backup programs.

Scheduled backups are essential in any production shop. Defining them is simple and flexible. You can run daily, weekly, monthly and yearly backups. Periodic backups are defined in terms of savepacks and tape pools. For unattended backup, you can require a new tape with each backup.

Pricing is reasonable compared to an NT server network backup package. The shareware version has no set price, but a quarter of contributions over $20 will be contributed to Software in the Public Interest, the organization which supports the Debian Linux distribution. Details are readily available on the Knox web site.

Using Arkeia on Linux means no expensive NT server licenses. It also means you can find a use for that old 486 box the IT department has been using for a doorstop because NT on anything short of a Pentium 200 is too slow to use.

Network backup is probably *the* mission-critical application in any organization. I have no qualms about recommending Arkeia on Linux for that application. If you want to investigate this package further, get the manual from Knox's web site and read it. Better yet, get the manual and the software and use them, as I did.

Charles Curley (http://w3.trib.com/~ccurley/) 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, WA. He can be reached via e-mail at ccurley@trib.com.

Archive Index Issue Table of Contents

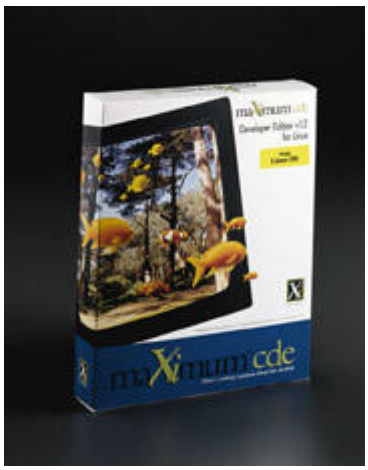
Advanced search

Xi Graphics maXimum cde/OS v1.2.3, Executive Edition

Jeff Alami

Issue #60, April 1999

The target market for maXimum cde/OS is enterprises wishing to run a commercially supported Linux distribution and the well-known CDE on their desktop workstations.



- Manufacturer: Xi Graphics
- E-mail: info@xig.com
- URL: http://www.xig.com/
- Price: $199.95 US; $349.95 US, Developer's Edition
- Reviewer: Jeff Alami

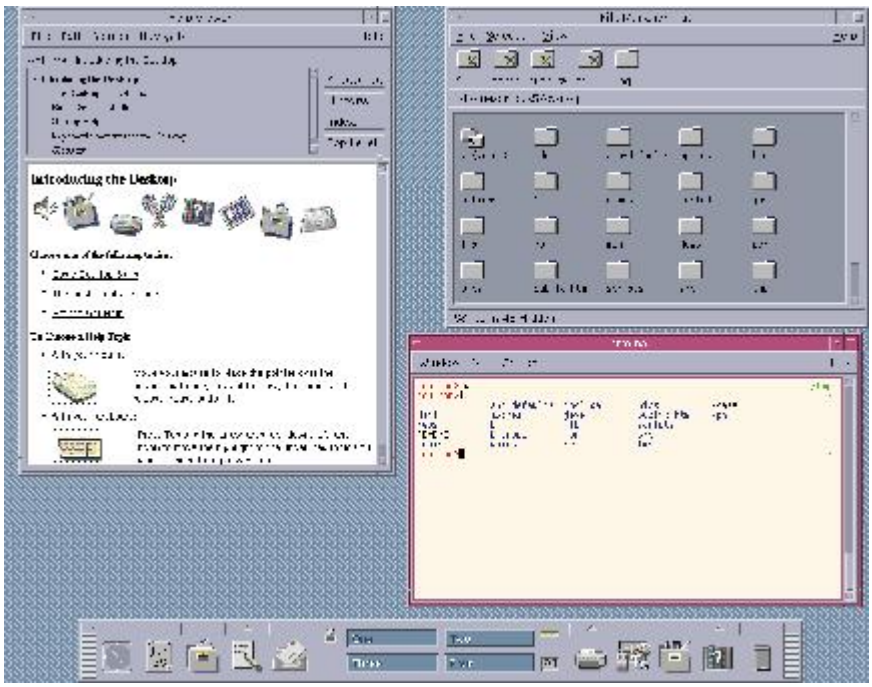
On July 10, 1998, Xi Graphics announced their decision to develop and sell their own distribution of Linux. Xi Graphics' maX*i*mum cde/OS incorporates Red Hat Linux 4.2, Xi Graphics' Accelerated-X Display Server, and the Common Desktop Environment (CDE) used in many commercial UNIX installations. The target market for maX*i*mum cde/OS is enterprises wishing to run a commercially supported Linux distribution and the well-known CDE on their desktop workstations.

I did not receive a boxed set from Xi Graphics, but simply a CD in a jewel case. My assumption is that they would most likely provide the same box and documentation as their CDE product, maX*i*mum cde. I installed the distribution on a Pentium 166 with 24MB RAM, a 4.3GB hard drive and an S3 Trio64 video card.

Installation

After creating the installation floppies, I booted from the floppies, which launched the installer program. The installer for maX*i*mum cde/OS is practically the same as Red Hat Linux 4.2. I used **fdisk** for partitioning, which can be difficult for a beginning user, but a competent UNIX system administrator should not encounter any difficulties. No choices were offered as to which packages were to be installed; the program simply installed 133MB worth of software. After installing the necessary packages, the installer ran the Accelerated-X registration and configuration program. The program did not autodetect my video hardware; I had to select the video card from a supplied list.

All things considered, the installation is not difficult for the experienced user. Given that the target market is not the consumer, this may be warranted, much like the installation for Caldera OpenLinux. In contrast, the latest version of Red Hat Linux has an easier installation, with automatic disk partitioning setups, easy package choices and video autodetection.



The Common Desktop Environment

Configuration

The Accelerated-X configuration program allows the user to choose which video card, input device and monitor are to be used. If you prefer editing text files, the configuration information is available in the file /etc/Xaccel.ini. Configuration for CDE is generally quite difficult compared to other desktop environments available for Linux. Reading through the CDE manual is recommended to understand how applications are added and how the environment can be configured.

Most of the other configuration features are inherited from Red Hat Linux 4.2. The control panel allows for user configuration, time and date, printer, network, modem and package configuration.

Features

Xi Graphics maX*i*mum cde/OS features the Accelerated-X Display Server and maX*i*mum cde, Xi Graphics' CDE implementation. Other featured software included in the package are Adobe Acrobat Reader 3.0, ImageMagick, Netscape Communicator 4.05, the Xanim movie viewer and the xmcd CD player. These applications are integrated into the CDE desktop.

Documentation and Support

I didn't receive any documentation, so I cannot comment on its quality. However, good sources of documentation would include the maX*i*mum cde manual (for learning how to use CDE) and the Red Hat Linux 4.2 User's Guide, available at Red Hat Software's web site.

Technical support for installation of maX*i*mum cde/OS is included at no charge for 30 days after registration. Corporations can purchase additional support packages from Xi Graphics. Of course, you can always go to traditional channels of Linux support, including newsgroups and mailing lists. Since it is so similar to Red Hat 4.2, many Linux users will be able to provide support.

Conclusion

At under $200, maX*i*mum cde/OS is sold at the same price as maX*i*mum cde without the Linux distribution. I would suggest getting the product with the operating system for two reasons. First, the Linux distribution included has been configured and tested with CDE. Second, you can still install their display server and CDE product on another distribution, from the /cde directory on the distribution CD. Of course, license restrictions still apply.

Xi Graphics has provided an excellent product for corporate UNIX users wanting to take advantage of Linux without a steep learning curve. Most commercial UNIX administrators are familiar with CDE on desktop workstations, and expect an integrated and supported package as provided in maX*i*mum cde/OS. The Accelerated-X Display Server is also a benefit for graphical workstations, as it provides better speed than XFree86 for many video cards. Of course, maX*i*mum cde/OS does have its limitations, namely the lack of glibc support found in the latest Linux distributions.

Jeff Alami is the director of Guardian Consulting Services, a consulting company specializing in Linux-based e-commerce and accounting solutions. He also writes Linux articles for 32 Bits On-line Magazine in his spare time. Jeff can be reached via e-mail at jalami@gcs.bc.ca.

Archive Index Issue Table of Contents

Advanced search

Book Review: *LINUX for Dummies, Quick Reference, 2nd Edition*

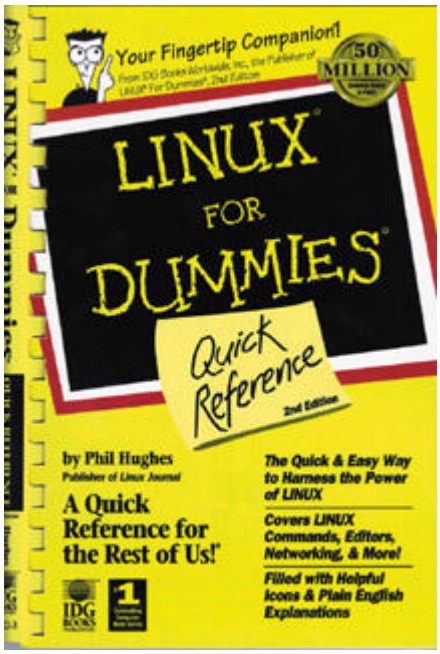
Harvey Friedman

Issue #60, April 1999

LINUX for Dummies, Quick Reference, 2nd Edition, by Phil Hughes

LINUX for Dummies, Quick Reference, 2nd Edition

- Author: Phil Hughes
- Publisher: IDG Books Worldwide, Inc.
- Price: $15.00 US
- ISBN: 0-7645-0302-2
- Reviewer: Harvey Friedman



I reproduce here the first two paragraphs from the introduction to this book because I think the author succeeded in doing exactly what he described:

> *LINUX for Dummies Quick Reference* fills you in on some of the basic capabilities of the Linux system, along with an assortment of those that aren't so widely used—commands and options you would discover only after years of experience using Linux.
>
> Although this book isn't the whole story on Linux, I have tried to provide you with the most amount of useful information possible in a compact reference. For instance, descriptions of Linux commands include examples of their usage to illustrate how Linux syntax really works.

This quick reference is not a quick reference to *Linux for Dummies* but to the Linux system, so it is not organized in the same way. I'll try to summarize what is in it. There are ten parts, an appendix of on-line resources and a glossary. Part I, “Getting to Know Linux”, covers what Linux is, selecting a Linux flavor, installing Linux, and fixing installation problems.

Part II, “Understanding the Shell”, covers available shells, basic directory commands, character quoting, command history, customizing the environment, directory naming conventions, directory referencing, files associated with a program, file naming conventions, path names, shell command elements, shell variables, special characters and startup files.

Part III, “Common Shell Commands”, covers commands for bash and/or the Korn shell but not csh, tcsh or any other shell. Luckily, most of the commands described are common to these shells.

Part IV, “Using X/FVWM”, covers an introduction to X, the FVWM Desktop Anatomy, adding backgrounds, checking out programs under X, examining pull-down menus, exiting X, modifying window characteristics, mousing with X, moving around the screen, starting applications, starting X, switching tasks, using button bars and using keyboard shortcuts.

Part V, “Text Editors and Working with Text”, covers the editors joe, Pico, Emacs, vi, spell-checking with ispell and formatting with fmt and groff.

Part VI, “Sending and Receiving E-Mail”, covers understanding the pieces of an e-mail system, using Elm and Pine (the most popular screen-based e-mail programs), getting your e-mail remotely with POP and working with metamail files.

Part VII, “Working with the Other Guys”, covers working with MS-DOS files and media and working with Macintosh media, working with UNIX files and media and converting data using Linux utilities.

Part VIII, “Networking”, covers transferring files over a network, working interactively on remote machines and checking network connectivity.

Part IX, “System Administration”, covers adding users and groups, connecting to an ISP with PPP, starting and stopping the system, kernel modules and performing system maintenance.

Part X, “Using Regular Expressions”, covers understanding simple regular expressions and combining expressions, and looks at some examples.

A few awkward explanations I found while reviewing the first edition were corrected in this second edition, using my suggested phrasings. I have to admit I missed several other minor problems which I list below:

- On page 20 in the example, both “super” and “stewart.txt” refer to the same file so should be the same.
- On page 66, it would be nice to see an example of removing strangely named files, as this is tricky if one has not seen it done.
- On page 107, “:rangess/...”; should be replaced with “:ranges/...”; meta-notation in an example is confusing.
- On page 198, there is a simple typo under LILO; “than” should be “that”.

Since I didn't notice those when I reviewed the first edition, there may be more that remain to be fixed.

I expected the Quick Reference to be good because I have used some of the pocket references Phil Hughes wrote for SSC, and I wasn't disappointed. I let my wife, who uses an SGI machine at work, look at it and she concluded that this would be a good reference for UNIX generally, not just for Linux.

It has been my impression that any of the *Dummies* series of books should lead an uninformed user in a step-by-step didactic fashion, possibly using some humor to make difficult concepts more accessible. I think this is one of the better *Dummies* books, as it is factually correct and entertaining as well. Of course, those who grew up using a GUI and don't deal with command-line interfaces may have to wait for the video.

Harvey Friedman is a computer consultant at the University of Washington, functioning either as system administrator or statistical analyst. In his leisure time, he likes playing with Linux and enjoys orienteering, the sport of navigation. He can be reached via e-mail at fnharvey@u.washington.edu.

Archive Index Issue Table of Contents

Advanced search

Conix 3-D Explorer

Michael J. Hammel

Issue #60, April 1999

This software, from Conix Enterprises, is an add-on package to Mathematica that allows you to use OpenGL 1.0-based rendering features directly from within the Mathematica environment.

- Manufacturer: Conix Enterprises, Inc.
- E-mail: info@conix3d.com
- URL: http://www.conix3d.com/
- Reviewer: Michael J. Hammel

Most likely, many of you know about the large number of graphical features available in Mathematica (see “Mathematica version 3.0 for Linux” by Patrick Galbraith, *Linux Journal*, December 1998), the Swiss army knife of mathematics software. While its features are fairly good for quite a few situations, I have always felt it would be nice to combine the power of Mathematica with an OpenGL-based 3-D interface. Such a connection would offer interesting possibilities for 3-D demonstrations.

Enter 3-D Explorer. This software, from Conix Enterprises, is an add-on package to Mathematica that allows you to use OpenGL 1.0-based rendering features directly from within the Mathematica environment. A simple installation process allows you to have the software up and running in only a few minutes and provides endless possibilities.

Since I am not a Mathematica expert, my goal for this review was to find out how easy it is to install and get started with both Mathematica and Conix 3-D Explorer. I also set out to verify that the examples provided were both understandable and functional. Finally, I wanted to see if it was possible to

create anything interesting with these two pieces of software in the short amount of time I had to write the review.

Installation

3-D Explorer comes packaged on two 3.5 inch floppy diskettes. The first diskette is the 3-D Explorer add-on for Mathematica; the other is a set of OpenGL libraries for use with 3-D Explorer. Installation instructions with the software were pretty basic—a single sheet of paper. I checked the Conix web site at http://www.conix3d.com/, and although it contained many samples and other information, nothing was available on how to read the diskettes. Overall, the web site does not offer much help specific to the Linux user.

Since the format of the diskettes was not specified, I relied on my UNIX experience and the knowledge that many commercial distributors seem to like the DOS format. I mounted the floppies as DOS diskettes. It worked. You can mount the diskettes with a command that looks like this:

```
mount -t msdos /dev/fd0 /mnt/floppy
```

The device you use may differ (fd1 for a second floppy drive, for example) and the mount point, /mnt/floppy, can be any existing directory that is empty. Once mounted, you can list the contents of the diskettes to see what is contained in the gzipped tar file. The installation file name on the first diskette, 3dexp101.tgz, does not match the instructions (3-DExplorer_1.0.tar.gz), but since only one installation file (plus two text files) is on that diskette, it is not hard to figure out.

A quick check of contents of the installation file shows that relative paths are used. That is, the files within the gzipped tar file do not include absolute directory paths. Unpack this file using the commands:

```
cd /usr/local/mathematica/AddOns/Applications
tar xvzf /mnt/floppy/3dexp101.tgz
```

This will unpack the 3-D Explorer files under the default Mathematica AddOns directory. Doing this guarantees that Mathematica will be able to see 3-D Explorer when Mathematica starts up.

After you have installed the first diskette, you can start Mathematica to view the on-line documentation. Help on using GLExplorer is available from the Mathematica Notebook Help browser. Run **Rebuild->Help Index** to get access to this browser as the last step of the installation process. The on-line documentation states:

> To install GLExplorer on a UNIX system, you must have a functional installation of OpenGL. On Linux systems, GLExplorer comes with OpenGL for Linux by Conix, but other OpenGL implementations may also be used.

Three things must be noted here: the first is that the on-line documentation does not refer to the package as 3-D Explorer but rather GLExplorer, so I will use the two terms interchangeably. Second, I am not sure GLExplorer works with other OpenGL implementations without installing the OpenGL package (the second diskette) from Conix. Third, since it does not appear that the Mesa libraries work (I did not try with my Xi Graphics OpenGL distribution), you must exit Mathematica and install the Conix OpenGL diskette before continuing.

In my first attempt to work with 3-D Explorer, I tried to use the Mesa libraries already installed on my system under /usr/local/lib, but these did not seem to work. The problem may be that my Mesa installation is not quite up to snuff; however, when I installed the Conix-supplied OpenGL package, I found a slew of files I would not have expected with a standard OpenGL distribution. For example, a directory called /GL was created under /usr/X11R6/lib which contained shared objects such as GLEngineClient.so.1 and GLRendererRGB565A8D32.so.1. A quick check with **ldd** on the executable under /usr/local/mathematica/AddOns/Applications/GLExplorer/GLLink.exe/Linux provided the following information:

```
libGLU.so.1 => /usr/local/lib/libGLU.so.1
libGL.so.1 => /usr/X11R6/lib/libGL.so.1.0
libXext.so.6 => /usr/X11R6/lib/libXext.so.6.3
libX11.so.6 => /usr/X11R6/lib/libX11.so.6.1
libdl.so.1 => /lib/libdl.so.1.7.14
libm.so.5 => /lib/libm.so.5.0.6
libc.so.5 => /lib/libc.so.5.3.12
libGLClientSys.so.1 => /usr/X11R6/lib/libGLClientSys.so.1.0
libXintl.so.6 => /usr/X11R6/lib/libXintl.so.6
```

Since libGLClientSys.so.1 is contained in the OpenGL distribution from Conix, it would appear you do indeed have to install that package in order to properly use 3-D Explorer.

The OpenGL diskette needs to be unpacked from the root directory, for example:

```
cd /
tar xvzf /mnt/floppy/cnxgl140.tgz
```

Again, the file name on the diskette does not match the written instructions on the single sheet of paper that comes with the diskettes. Additionally, the single sheet of instructions says to run **ldconfig** after you install the OpenGL files. However, the files in the gzipped tar file on the diskette use relative paths that start with usr/X11R6/lib, so you do not have to do this as long as you first change to the root (/) directory.

Registration and/or license IDs are not required in order to use this product. Once the OpenGL package has also been installed, you are ready to try out GLExplorer. Note that Mathematica uses Motif, but the version I received for this review was statically linked—Conix's package does not require Motif to work.

Documentation

The only documentation for GLExplorer is the on-line manual. Even if you have never used Mathematica, their help system is quite easy to use and includes hypertext links with expandable/retractable sections. You can execute the example commands listed in the Help browser by clicking on the cell bracket (at the right of each cell) with the right mouse button and selecting **Kernel->Evaluation->Evaluate Cells**.

All example uses of GLExplorer covered in the on-line documentation require that you first run the command

```
Needs["GLExplorer`GLRenderer`"]
```

This command starts the GLExplorer rendering engine. Once started, a small icon (see Figure 1) will appear on your desktop. Although this command is listed at the start of each example, you need to run the command only once to start the engine. After that, you can skip this command in each of the subsequent examples you try.

Figure 1. GLExplorer Engine Icon

Running Mathematica/3-D Explorer

The first thing I noticed about running Mathematica, on its own, was the dialog box that pops open at start time. This box complained about the fonts not being installed correctly, but if I clicked on the Continue button in the dialog everything seemed to work fine. Fonts necessary to run Mathematica are installed under $MATHEMATICA/SystemFiles/Fonts/Type1 and $MATHEMATICA/SystemFiles/Fonts/X, where $MATHEMATICA is the top-level directory where you installed all the Mathematica files. The default for this is /usr/local/Mathematica. For everything to work right, the Type1 fonts should be listed prior to the X fonts in your X server's font path.

While looking at the GLExplorer Demos provided, you will find reference to **GLShow**, a GLExplorer command. Conversion of Mathematica's 2-D and 3-D graphics into OpenGL-based graphics is handled through this command. The

command accepts standard Mathematica **Show** command syntax, making it fairly easy to make use of OpenGL for interactive graphics display without having to learn a lot about it.

GLShow immediately opens a new window. This window is small, about 256x256 pixels, but is interactive. You can immediately rotate any 3-D shape and translate (move around the window) any 2-D or 3-D graph.

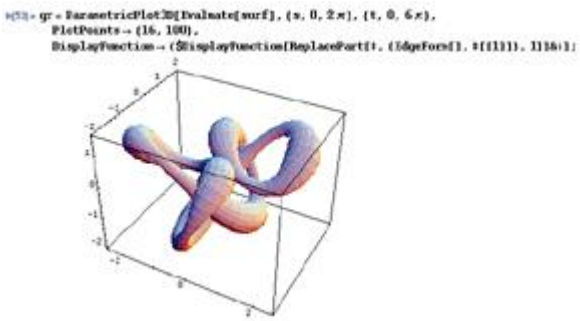


Figure 2. Snake Knot Demo

The Snake Knot demo shows how a complex set of formulas can be combined to produce interesting graphics with both Mathematica and GLExplorer. The command

```
gr=ParametricPlot3-D[Evaluate[surf],{s,0,2PI},{t,0,6PI},
PlotPoints->{16,100}, DisplayFunction ->
($DisplayFunction[ReplacePart[#,{EdgeForm[],#[[1]]},1]]&)];
```

produced the Mathematica graphic shown in Figure 2. Plugging this into GLShow and setting a number of optional parameters for it,

```
GLShow[gr,ShadeModel->Smooth,DepthCompression->False,
   Axes->False,AdditionalLights->
   {{{0,-1,-1},RGBColor[0,0,.8]},
   {{-.5,.5,-1},RGBColor[0,.8,0]}}];
```

produced the image in Figure 3. Note the values **PI** and **->** are used to represent text that only Mathematica's special fonts can show. Figure 4 shows the same object rotated and enlarged in the GLExplorer window. Rotations can be done using simple mouse-button clicks and drags. Enlarging the window will automatically redraw the object it contains to fit the new window size.

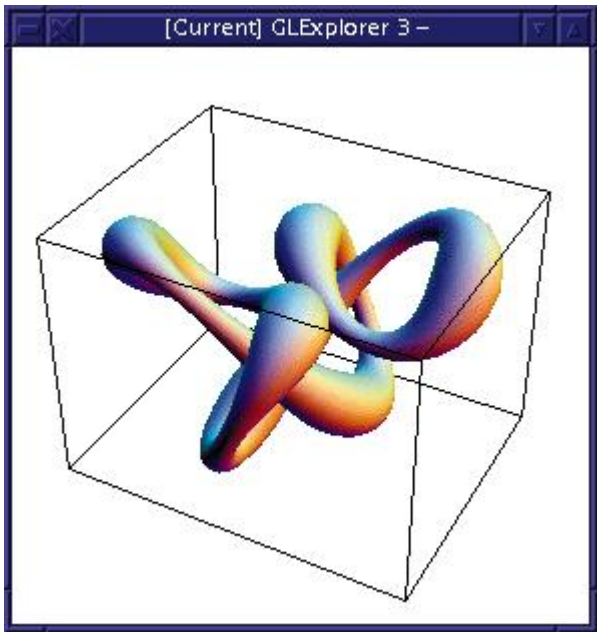


Figure 3. GLShow Display

Apparently, the windows opened by GLShow can be closed only by using the window manager's window menu Close option. If there is a more appropriate way to close the windows, I could not find it.

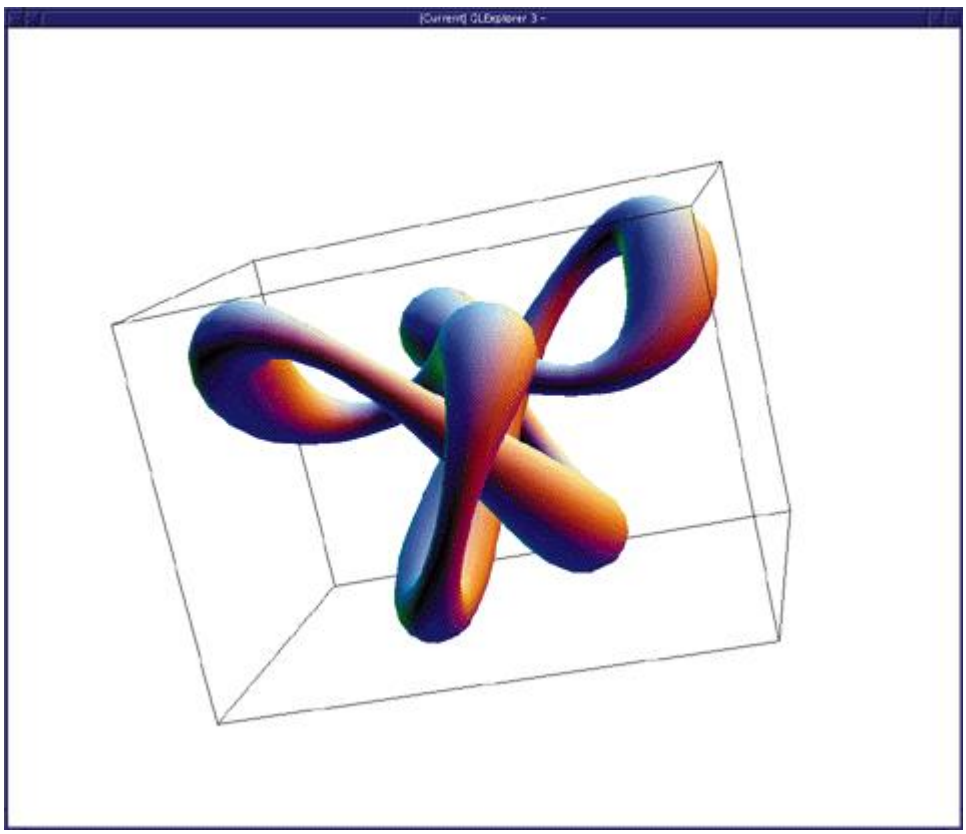


Figure 4. Knot Rotated and Enlarged

OpenGL 3-D Support

3-D Explorer is not a complete implementation of OpenGL. Rather, it offers an interface that allows direct access to a subset of OpenGL, plus a method of extending this access to encompass any OpenGL function. In order to use the existing or extended set of OpenGL primitives and directives, you need to understand the 3-D Explorer GLGraphics object.

Supported features of OpenGL are those expected for any good 3-D tool. They include drawing and color primitives, antialiasing, lighting and transformations. Accessing these primitives is done with the **GLGraphics** object. The GLGraphics object allows you to specify associations of points, lines, objects, colors, etc., all of which can then be viewed using GLShow.

Drawing primitives supported by GLExplorer include (but are not limited to) points, lines, triangles, polygons, surfaces and cuboids. Color primitives include RGB/RGBA, CMYK, HSL and gray-scale colors. All drawing primitives can have their sizes (thickness) and colors set using 3-D Explorer directives such as PointSize or EdgeForm. Antialiasing, hidden line removal and lighting are all supported. In fact, any OpenGL capability recognized by the OpenGL **glEnable** function can be set using the Enable directive of GLExplorer.

Transformations such as rotation, scaling and translating are supported locally by GLExplorer. These are handy when used with OpenGL display lists. Display lists are a method of grouping OpenGL primitives together so they can be rerun as a single object. This method of grouping provides better efficiency and is part of the OpenGL specification itself. GLExplorer offers complete access to this feature.

Texturing of OpenGL objects is supported; however, the demo showing a texture on a wave (from the on-line documentation) crashed GLExplorer. I was never able to run this example successfully.

You can extend GLExplorer to encompass any OpenGL function through the use of the **Commands** wrapper. This GLExplorer command allows you to call any OpenGL function from within the format of a Mathematica graphics expression.

Programming

OpenGL programs can be written directly within Mathematica with the help of GLExplorer. All of the OpenGL core functions are accessed with the prefix gl, such as **glBegin[GlLineStrip]** or **glVertex[0,0]**. Similarly, the OpenGL Utility library functions are accessed with the glu prefix. This is just as you would use them in any other OpenGL application written in C, for example (although C syntax

differs from Mathematica's—the function names are the same). In the case of windowing commands, GLExplorer prefixes commands with glM, such as **glMCreateWindow[]** or **glMGetWindowOptions[glwin,ErrorTrapping,ImageSize]**.

The section in the User's Guide on using these direct access commands to OpenGL is limited to defining their use. It leaves the explanation of what they are used for to the canonical texts on OpenGL from Addison Wesley, which are quite good. If you intend to become more familiar with OpenGL for use with Mathematica and GLExplorer, I highly recommend these books.

Summary

Overall, the speed of GLExplorer's rendering engine is very good. I use a Cyrix 200 with 64MB of memory and none of the examples in the on-line manuals took more than a few seconds to generate. They also reacted interactively quite well—interactive rotation and translation of the displayed images was smooth and immediate. This is all done with software acceleration; no 3-D hardware acceleration was used. It should be noted, however, that few lights were used in the examples and the distributed OpenGL libraries could not be compared to similar software-accelerated libraries from Mesa or Xi Graphics.

GLExplorer offers the user a method of building an OpenGL-based image from a command line, step by step. You can even use Mathematica to write an OpenGL program and run it directly from within Mathematica, then let GLExplorer handle the image display and interaction for you.

In setting out to write this review, I wanted to discover if a novice user such as myself could get up and running fairly quickly. As it turns out, with the fairly good 3-D Explorer documentation and Mathematica's terrific Help browser, I was able to find my way around both tools quite easily. If you are looking for a way to integrate your Mathematica notebook graphics with OpenGL, Conix 3-D Explorer may just be your ticket.

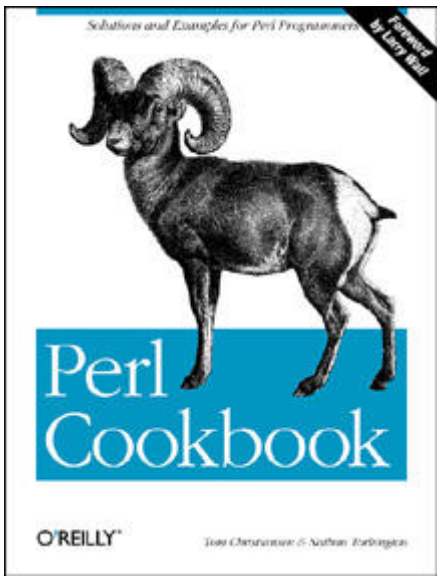
Michael J. Hammel (mjhammel@graphics-muse.org) is a Computer Science graduate of Texas Tech University, and a software developer specializing in X/Motif. Michael writes the monthly “Graphics Muse” column in Linux Gazette, maintains the Graphics Muse web site and the Linux Graphics Mini-HOWTO, helps administer the Internet Ray Tracing Competition and recently completed work on his new book *The Artist's Guide to the GIMP*, published by SSC, Inc. His outside interests include running, basketball, Thai food, gardening and dogs.

Perl Cookbook

James Lee

Issue #60, April 1999

In this book, Tom Christiansen and Nathan Torkington cover many topics providing numerous code examples known as recipes (it is a “cookbook”, after all).



- Authors: Tom Christiansen and Nathan Torkington
- Publisher: O'Reilly & Associates
- E-mail: info@oreilly.com
- URL: http://www.oreilly.com/
- Price: $39.95 US
- ISBN: 1-56592-243-3
- Reviewer: James Lee

Having written programs in Perl for a number of years, I fondly remember spending many hours leafing through the first edition of O'Reilly's *Programming Perl* (also known as “the Camel Book”), an excellent book covering the Perl programming language up to version 4.0.36. That book was

well-written, technically correct and very witty. Two chapters of the first edition of *Programming Perl* I was particularly fond of, “Common Tasks with Perl” and “Real Perl Programs” contained numerous common tasks and real programs using Perl.

In 1996, O'Reilly published the second edition of *Programming Perl*, an excellent book covering all features of Perl including those features new to Perl version 5. The size of the book expanded from 465 pages in the first edition to a whopping 645 pages in the second. Many topics new to Perl 5 were added to the second edition, including a discussion of references and object-oriented programming. However, due to its size, portions of the first edition could not be covered in the second. Therefore, parts of the first edition were deleted, including my favorite “Common Tasks with Perl” and “Real Perl Programs” chapters.

Perl Cookbook (also known as “the Ram Book”) is a companion to *Programming Perl*, expanding the chapters of real Perl programs in the first edition of the Camel book to a massive 757 pages of thoroughly explained, useful Perl code. In this book, Tom Christiansen and Nathan Torkington cover many topics providing numerous code examples known as recipes (it is a “cookbook”, after all). The introduction claims it is not a tutorial nor a reference, but rather states “this is a book for learning *more* Perl”. And so it is.

This is a book for all types of Perl programmers from beginner to expert. If you are looking for a book with hundreds of examples to cut and paste, or something that explains in detail how to do many different things using Perl, this book is for you.

The book starts with recipes to handle Perl's basic data types. Each of the following five topics gets its own chapter: strings, numbers, dates and times, arrays and hashes.

Chapter 6 discusses pattern matching and goes into great detail on using Perl's regular expressions.

Chapters 7, 8 and 9 deal with file access, file contents and directories. Chapter 10 discusses subroutines (check out the nifty programs to sort e-mail at the end of this chapter).

Chapters 11, 12 and 13 discuss references, packages, libraries, modules, classes, object and ties. These topics are traditionally difficult to master with Perl, yet I found these chapters to be well-written and understandable. I was particularly impressed with the section on using **tie** at the end of Chapter 13.

Chapter 14 covers database access, including UNIX DBM databases and the DBI module. Chapter 15 discusses user interfaces from the terminal, the Tk interface and the Expect programming language.

The remaining five chapters discuss networking with Perl. Topics include process management and communication, sockets, Internet services, CGI programming and web automation.

I have found this book to be extremely helpful, technically excellent and loaded with useful and usable source code. For example, I recently wrote a CGI program that accepts a floating-point number from the user, and I wanted to use a regular expression to verify that the number entered was valid. I thought to myself, “Hmm, I can either create the regular expression myself, or see if this issue is covered in the *Perl Cookbook*.” Noting that “Laziness” is one of the three primary virtues of a Perl programmer, I decided to look in the book. In the table of contents, I immediately found a recipe titled “Checking Whether a String Is a Valid Number”. Sensing I was on to something, I turned to page 44 and found the following regular expression to verify correct C-style floating point numbers:

```
/^([+-]?)(?=\d|\.\d)\d*(\.\d*)?([Ee]([+-]?\d+))?$/
```

I suppose that, with a bit of time and a lot of testing, I could have come up with a working regular expression. But I didn't need to—the answer was found in the *Perl Cookbook*.

The *Perl Cookbook* has the answers to many common Perl questions. Need to find all unique entries in a list? Try page 102. Want to create a hash, but retrieve the key/value pairs from the hash in the order entered? Turn to page 139. Want to POP your e-mail from your server? See page 656.

I especially liked Chapter 20, entitled “Web Automation”. Did you know it takes only five lines of Perl code to submit a form to a CGI program using the **GET** method? This example is from page 710:

```
use LWP::Simple;
use URI::URL;
my $url =
url('http://www.perl.com/cgi-bin/cpan_mod');
$url->query_form(module => 'DB_File', readme => 1);
$content = get($url);
```

Mirroring a web page takes only two lines of code (from page 724):

```
use LWP::Simple;
mirror($URL, $local_filename);
```

Code like this makes Perl the world's most useful programming language. Examples like these make the *Perl Cookbook* the most useful Perl book I own.

The book contains a mountain of source code, all of which is available from the O'Reilly FTP site (ftp://ftp.oreilly.com/published/oreilly/perl/cookbook/). There, you can find over 130 full-length programs as well as all of the code snippets from the book.

Since this is a cookbook covering a number of topics, readers looking for in-depth discussions of the topics mentioned above should be aware of the fact that they are not covered in minute detail. For instance, the chapter on CGI programming is a brief description that will be excellent if you have some knowledge of CGI, but if you are looking for in-depth discussion, you should check out a CGI book. This makes sense—cookbooks are not tutorials. They present recipes, with the assumption that you know the basics.

In summary, if you are a Perl programmer, this book is essential. If you are new to Perl and want to become a Perl programmer, this book is highly recommended as a tool to learn more Perl. If you are wondering what Perl is all about and what Perl can do for you, this book deserves a look. If you haven't heard of Perl, where have you been?

James Lee is the president and founder of Onsight (http://www.onsight.com/). When he is not teaching Perl classes or writing Perl code on his Linux machine, he likes to spend time with his kids, root for the Northwestern Wildcats (it was a long season), and daydream about his next climbing trip. He also likes to receive e-mail at james@onsight.com.

Archive Index Issue Table of Contents

Advanced search

grep: Searching for Words

Jan Rooijackers

Issue #60, April 1999

A command to help you find a specific word or a sentence in a file.

Within Linux (or any other UNIX), many people make use of filters, small programs (black boxes) that read input from standard input (stdin), do something with this input, and return the result to standard output (stdout).

Linux has many filters. Some examples are:

- **wc**: print the number of bytes, words and lines in a file
- **tr**: translate or delete characters
- **grep**: print lines matching a pattern
- **sort**: sort lines in a file
- **cut**: cut selected fields from a file

The easiest way to learn these filters is to use them. This may seem daunting at first, since you may not know all the capabilities of these filters. I will describe the functions of grep so that you can benefit from its power.

I will be using this article (article.txt) as the input file for all the examples.

The Syntax

The syntax of the grep command is as follows:

```
grep [ -[[AB] ]num ] [ -[CEFGVBchilnsvwx] ]\
[ -e ] pattern| -file ] [ files... ]
```

I use GNU grep Version 2; if you're using another version, you may have slightly different options. I will touch on only those options I use most. To learn more about the grep command, see the man page. Variants of the grep command are **egrep** and **fgrep**. **grep** includes flags to simulate these commands: **-E** for egrep and **-F** for fgrep.

The simplest form of the command is:

```
grep flip article.txt
```

This will search for the word “flip” in the file article.txt and will display all lines containing the word “flip”.

grep also accepts regular expressions, so to search for “flip” in all files in the directory, the following command can be given:

```
grep flip *
```

All lines in all files which contain the word “flip” will be displayed, preceded by the file name. Thus, the first line of the output will look like this:

```
article.txt:grep flip article.txt
```

The line begins with the name of the file containing the word “flip”, followed by a colon, then the appropriate line.

Sometimes you may want to define the search for special characters or a word combination. To do this, put the expression between quotes so that the whole expression/pattern will be treated as one. The command would then look like this:

```
grep -e "is the"
```

I put the **-e** (i.e., do pattern search) option in this example just for demonstration purposes. It is not necessary to specify, as it is the default value.

To see the line numbers in which the pattern is found, use the **-n** option. The output will look like that shown above, with the file name replaced by the line number before the colon.

Another option which provides us with a number is the **-c** option. This option outputs the number of times a word exists in a file. This article contains the word “flip” 10 times.

```
> grep -c flip article.txt
10
```

grep and speed

You may now be able to think of many ways in which you might use grep. For any command you use often, speed is important. Normally, grep can do its job quickly. However, if the search is being done over many large files, the results will be slower to return. In this case, you can speed up the process by using

either **fgrep** or **egrep**. **fgrep** is used only for finding strings, and egrep is used for complicated regular expressions.

Conclusion

File names, words, sentences and numbers can all be found quickly using grep. In addition, using the grep command together with other filters can be very powerful and prove to be of great value. For example, you could search a statistics file and sort the output by piping it through the **sort** and **cut** commands (see man pages):

```
grep ... | sort ... | grep ... | cut ... > result
```

This has been a quick introduction to get you started and rouse your curiosity to learn more about grep and other filters.

Jan Rooijackers works for Ericsson Data Netherlands (DSN) as an IT engineer. One of his favorite hobbies is programming (Tcl/Tk) and trying out new things in the computer world. He spends as much time as he can with his wife and two sons. He can be reached at dsnjaro@apskid.ericsson.se.

Archive Index Issue Table of Contents

Advanced search

Linux 2.2 and the Frame-Buffer Console

Joseph Pranevich

Issue #60, April 1999

Wondering about the new frame-buffer features in the kernel? Mr. Pranevich gives us the scoop.

Linux is a fascinating and fast-paced beast. It seems like only yesterday the hardy developers of the Linux kernel were busily putting the finishing touches on ELF support, loadable modules and SMP (Symmetric Multiple Processing)--things we take for granted today. In those days, more time was spent on the critical hurdles, the ones that would turn Linux into the great server platform it is today. In today's Linux world, more time is spent on the less critical components of the system: new file systems, television and radio cards and parallel-port drives. I feel the increased interest in the operating system by the world's corporations will make even more “non-critical” hardware be supported in the future.

Linux 2.2 is a milestone for Linux's development. No longer is Linux a niche operating system—it is a viable solution for the masses. With support for so many new options, it's no wonder.

Part I. What We Have Today

Text-mode, Linux's most basic output method, is also one area where Linux has changed little since the “old” days. The text-mode console of Linux 2.0 is pure in its simplicity with one obvious (and maybe striking) fact: no code for graphical primitives is in the kernel. The lowest level interface to the text-mode kernel is as simple as the file stream. Higher level functions, such as would be required for full-screen text console applications (Pine, etc.), are done through a superset of the vt100 terminal protocol. Libraries, such as ncurses, are built on top of this to simplify programming and to provide a sort of terminal abstraction. Applications written for Linux's text-mode console using this abstraction can run on just about any terminal. Scroll back and Linux's famous virtual consoles are not a sophisticated extension of those basic building

blocks, but rather an extension provided because of the text-mode driver's close relationship to the VGA hardware on which it was designed.

Other utilities exist, such as SVGATextMode, which allow us to access some of the graphical capabilities of hardware in text mode. These utilities generally allow one to change text resolution, fonts and colors. However, these extensions directly interface VGA features and do not call on any extensions in the kernel. Generally, the Linux console is so modular in design that it does not notice the low-level VGA feature changes. These utilities should not be confused with SVGALib or the frame-buffer console, as they allow manipulation only within the hardware's text modes.

SVGALib (SVGA Library)

There is obviously quite a bit more to modern hardware than simple text modes. As mentioned before, kernel hooks are not provided for accessing these functions, but many user-space programs and libraries are available that bypass the kernel altogether to access the hardware beneath. (“User space” is a term used to describe the limited and protected “space” in which user programs run. In contrast, “kernel space” routines are generally unprotected and can cause crashes and other problems. User protections can be removed in order to allow user programs to access hardware.) Unfortunately for SVGALib, the most popular alternative, the support API provided is heavily tied to the features of the VGA hardware, making it difficult to port either the SVGA library or the end application to any other type of hardware. The other downside is that this library does not support all VGA hardware at its full potential, but that can be forgiven due to the turbulent nature of hardware design. Limitations aside, SVGALib has proven to be a stable and popular solution to the console problem and is the primary interface used in Quake and other games.

The X Window System

The final and most popular option for accessing the video hardware under Linux is through the X Window System, the most common GUI subsystem for UNIX. The X Window System includes an “X Server” which is similar in purpose to a Windows-style video driver. In addition to “driver” features, the X server includes code for running programs over a network and handling many GUI tasks internally. In this respect, the “driver” portion is not truly separate from the “server” portion. A program that wishes to access the video hardware would do so by communicating with the X server through its API.

The first and most obvious disadvantage of this approach is that it would be difficult, if not impossible, on some setups to run a “console” (full-screen) application via this method. Second, because of the combined driver/server

features of the X system, servers tend to be very large, making it difficult to allow programs to run in low-memory situations. The third disadvantage of this approach is that it is fairly common for companies to profit by selling closed-source X servers for new hardware. However, the primary advantage of this system, having a long-standing and cross-platform pillar to base graphical applications on, seems to outweigh the negatives.

Non-i386 Linux

Thus far, I have managed to not even mention the role frame-buffers play in current Linux kernels. The frame-buffer console system has been a staple of several other UNIX systems for quite some time and several Linux flavors, in particular Linux/m68k and Linux/PPC. The issue on those platforms that made the frame-buffer console so important is a simple one: not all video hardware supports a built-in text-mode. On these systems, the frame-buffer console is not a new luxury which brings up a boot logo, but rather a requirement for functionality. On these systems, the picture is a tad different. They still have X, of course, and X is an excellent way of creating applications in a device-independent fashion. A special X server designed specifically for dealing with frame-buffer systems has nearly always been available. The picture on those systems is also simplified, since no library is available that would enable them to run SVGALib applications.

Security Implications

Under the current implementation, a number of security concerns must be raised when dealing with the graphics subsystem. It is partly because of these security implications that the frame-buffer console has become a part of the Linux 2.2 kernel.

Hardware access to user programs must be provided through a kernel API which provides a layer of device abstraction. To keep the system secure and crash-free, the kernel interface must be carefully designed, as it is within the kernel that the most damage can be done. Access to hardware is dangerous; it would be quite easy to crash the machine if a user program could access it directly. Also, badly designed kernel interfaces could reveal sensitive data about a user, breaches of security, no matter how difficult to exploit, are taken very seriously in the Linux developer community. While no kernel provisions exist for direct hardware access, there are certain workarounds which can be used to circumvent these restrictions—in a way that is as secure as possible.

There are two exceptions to the “no direct access” rule. The first is simple: with programs run by root, the system administrator (who should generally know what she is doing) can access any files and any hardware directly. In fact, there are virtually no limitations as to what root can do. But you certainly don't want

your users having the root password and using it each time they want to run Quake, do you? The second exception is the solution to this issue: the owner of a program (generally root) can set a flag called the “setuid bit” (Set User ID) on a program, allowing regular users to “be” the other user as far as the program is concerned. Thus, if root owns the file and it has the setuid bit set, that program will always have root access and therefore will always be able to access hardware directly. In this special and well-controlled case, end users can run their Quake or their X servers despite the fact that they normally wouldn't be able to access the hardware directly.

Note, however, that not all graphical programs will need to be “setuid root” to operate. X programs in particular do not access any hardware directly. Rather, they communicate to the X server via an API which then translates what they want to do onto the video hardware. SVGAlib applications, in contrast, generally all need to have special permissions to operate properly.

The security implications of having an arbitrary program run as root should be obvious. Linux does not currently have a method of saying that a particular program can “just” access any file or “just” access hardware directly—work is progressing in this area. Instead, a “setuid root” application can do anything, including shutting down the server. Talk about a denial of service! Thus, these applications must be *very* carefully written so that an errant user cannot do anything that would violate the security of the box. It is up to the administrator of the site to maintain and ensure these special programs are used only when absolutely necessary, otherwise gaping security holes will result.

Even a carefully written program can sometimes be cracked via “stack smashing” to gain root access if a programmer does not make sure to watch his buffers and use safer routines like **strncpy** instead of **strcpy**. In general, programmers who write “setuid root” applications should be aware of any buffers used throughout the program. For the record, strcpy and strncpy are both functions that copy text data (strings) from place to place in memory. The “n” in strncpy means there is a maximum number of characters to be copied. Otherwise, it would be possible for a cracker to manipulate data in such a way that the data being copied is larger than the place to which it is copied and the excess would overwrite memory. If they are skilled, this excess could include program code which would then be run to break into a shell or do other damage—and it would be executed as root.

Part II. Linux 2.2 Implementation

Under the hood, Linux 2.2's text mode was designed to be a more modular system with defined interfaces and less of a dependency on VGA hardware internals. The casual observer will not notice either of these improvements.

This is the same text mode we have come to love. However, improvements were made to allow better serial consoles for machines without video hardware and to allow for the turning off of virtual consoles, but these features are not expressly part of the text-mode driver.

Frame Buffers

Frame-buffer devices are the biggest new video feature of Linux 2.2 for i386. Unlike earlier versions of Linux, it is possible through the frame-buffer device to access the video hardware of a machine directly and in a device-independent manner. All the basic graphics primitives are supported, although acceleration is not generally supported at this time. Exactly how accelerated architectures will fit in is still a matter of debate. Some point to user-mode programs and libraries (such as GGI); others believe the best location for at least some types of acceleration is in the kernel. Note that, in contrast to the text-mode driver, there is no character cell display—it is handled elsewhere. SVGAlib applications and, to a lesser extent, X applications can both obtain approximately the same level of control over video hardware, but they are usually less portable and more of a security risk.

In addition to those features, it is now possible to access the frame-buffer devices through the use of device nodes, just like any other device. As an example, these device nodes, /dev/fd*, can be used to get a screen capture simply by doing a standard copy command.

Above the frame buffer is the frame-buffer console (fbcon). This is where the standard vt100+ terminal emulation is implemented. In fact, the emulation is so good that the end user may not even notice the system is in graphical mode. At this level, the kernel has a much larger control over fonts and other features formerly provided through utilities such as SVGATextMode.

Now, all of these features are great, but the major feature that will probably be the reason most Linux users try out the frame-buffer driver is the boot logo. Yes, now you can finally have a cute little picture of Tux carrying a beer whenever you turn on your computer.

Security Implications

Not nearly as many security problems exist under the new system as under the old. Programmers who write graphical applications do not need to be as careful as their legacy counterparts, since they are protected by the user-level security of the kernel. System administrators will also have fewer “setuid root” applications to track, making their security audits easier. In contrast, there will be more code in the kernel that could go awry, and kernel developers will need to keep the new video subsystem as bug-free as the rest of the Linux kernel.

Part III. Advantages of the Frame Buffer

My favorite feature of the new kernel subsystem, if it were to go into wide use, can be summed up in two words: cross-platform compatibility. Simply put, applications written to use the frame buffer will be immediately portable to all Linux platforms. This is in direct contrast to the current SVGALib system which does have some cross-platform compatibility—but only on systems with VGA-like hardware. This will, in theory, make compiling any graphical application on multiple Linux architectures as easy as compiling an application with ncurses is today.

Bear in mind, however, that this argument does not apply to X applications which are already cross-platform. Rather, this level of compatibility would help with X at a somewhat lower level.

Single X Server

Another advantage of having a single level of abstraction for video hardware at the kernel level is with X servers. Unlike older X servers which needed to concentrate on the specific features of a class of video cards, frame-buffer-aware ones can concentrate solely on the networking and other X aspects while allowing the kernel to handle video specifics. In my perfect world, that would allow developers more time to look at X issues rather than video issues. If a developer was still interested in developing video code, he or she could always look to the kernel for something to do.

The X server for the frame buffer is already available as Xfree86-FBDev and has shipped with some distributions.

Part IV. Disadvantages of the Frame-Buffer Driver

One of the largest downsides of the new kernel driver system, and the one that has been getting the most attention, is the question of stability. Will these new frame buffers impact the outstanding uptimes of modern Linux? The answer is simply no. One of the glorious things about Linux is that you are generally never forced to do things you don't want and disabling the frame buffer is as easy as a recompile (if the distributions even ship with it by default—something I highly doubt).

It is my opinion that Linux developers reach a higher standard than some other software developers do. (This might explain why I've never gotten anything more than teeny bits into the kernel myself.) Even in a worst-case scenario, a rampant kernel module is only somewhat more dangerous than a rampant “setuid root” X server. People accept that risk daily.

Lack of Drivers

The new driver system has less supported hardware than its legacy counterpart. Additionally, the supported hardware is generally in a sub-optimal or non-accelerated position. In contrast to XFree and to a lesser extent SVGAlib, that is an order of magnitude less support. The video subsystem today is a lot like the sound subsystem was yesterday, i.e., it supports very few cards. With time and patience, developers will no doubt make this new system as robust as possible.

Lack of Acceleration

Also, the frame buffer is not meant to be a generic base for an acceleration architecture. While it is likely that better acceleration will be provided in the future, we may have to wait until that becomes generally available. Alternately, it is quite likely that the GGI project (General Graphics Interface at http://www.ggi-project.org/) or some organization will propose and implement a workaround to this situation. Once again, due to the newness of the system, not all answers are immediate.

Part V. Other Notes

Red Hat 5.2 (which I use at home and at work) already includes support for the frame-buffer X server, FBDev. If you are experimenting with this feature of Linux 2.2 and have Red Hat, this will save you a download/compile cycle. Unfortunately, at this time there is no easy way to configure this device; I recommend consulting the documentation. Red Hat 6.0 may include this feature or make it easier to use.

VesaFB

To those who want to jump right into the frame-buffer console, an option exists that works with nearly all VGA-compatible video cards: the VesaFB driver. It does require a VESA 2.0-compatible BIOS. This driver is *not* a good example of what the new features of Linux 2.2 can do, but it does allow one to get her feet wet. In particular, it lacks support for resizing the screen resolution and it requires the mode to be changed at boot time. In reality, this driver is meant only as an example driver and your mileage may vary getting it into production work. Users of Linux on other platforms may have a better idea of how things should be in the end.

Joseph Pranevich (jpranevich@lycos.com) is an avid Linux geek and, while not working for Lycos, enjoys writing (all kinds) and working with a number of open-source projects.

LINUX JOURNAL

Writing Modules for mod_perl

Reuven M. Lerner

Issue #60, April 1999

Discover the flexibility and power of writing mod_perl modules instead of CGI programs.

CGI programs are a common, time-tested way to add functionality to a web site. When a user's request is meant for a CGI program, the web server fires up a separate process and invokes the program. Anything sent to the STDOUT file descriptor is sent to the user's browser, and anything sent to STDERR is filed in the web server's error log.

While CGI has been a useful standard for web programming, it leaves much to be desired. In particular, the fact that each invocation of a CGI program requires its own process turns out to be a large performance bottleneck. It also means that if you use a language like Perl where the code is compiled upon invocation, your code will be compiled each time it is invoked.

One way to avoid this sort of problem is by writing your own web server software. Such a project is a significant undertaking, though. While the first web server I used consisted of 20 lines of Perl, most servers must now handle a great many standards and error conditions, in addition to simple requests for documents.

Apache, a highly configurable open-source HTTP server, makes it possible to extend its functionality by writing modules. Indeed, modern versions of Apache depend on modules for most functionality, not just a few add-ons. When you compile and install Apache for your computer system, you can choose which modules you wish to install.

One of these modules is **mod_perl**, which places an entire Perl binary inside your web server. This allows you to modify Apache's behavior using Perl, rather than C.

Even if you plan to use approximately the same code with mod_perl as you would with CGI, it is useful to know that mod_perl has some built-in smarts that caches compiled Perl code. This gives an extra speed boost, on top of the efficiency gained by avoiding the creation of a child process in which to run the CGI program.

Over the last year, this column has looked at some of the most popular ways of using mod_perl, namely the Apache::Registry and HTML::Embperl modules. The former allows you to run almost all CGI programs untouched, while taking advantage of the various speed advantages built into mod_perl. HTML::Embperl is a template system that allows us to combine HTML and Perl in a single file.

Both Apache::Registry and HTML::Embperl offer a great deal of power and allow programmers to take advantage of some of mod_perl's power and speed. However, using these modules prevents us from having direct access to Apache's guts, turning it into a program that can handle our specific needs better than the generic Apache server.

This month, we will look at how to write modules for mod_perl. As you will see, writing such modules is more complicated than writing CGI programs. However, it is not significantly more complicated and can give you tremendous flexibility and power.

Keep in mind that while CGI programs can be used, often without modification, on a variety of web servers, mod_perl works only with the Apache server. This means that modules written for mod_perl will work on other Apache servers, which constitute more than half of the web servers in the world, but not on other types of servers, be they free or proprietary.

If portability across different servers is a major goal in your organization, think twice before using mod_perl. But if you expect to use Apache for the foreseeable future, I strongly suggest looking into mod_perl. Your programs will run faster and more efficiently, and you will be able to create applications that would be difficult or impossible with CGI alone.

Perl*Handlers

CGI programmers have a limited view of HTTP, the hypertext transfer protocol used for nearly all web communication. Normally, a server receiving a request from an HTTP client (most often a web browser) translates the incoming URL into the local file system, checks to see if the file exists and returns a response code along with the file's contents or an error message, as appropriate. CGI programs are invoked only halfway through this process, after the translation has taken place, the file has been found and a new process fired off.

mod_perl, by contrast, allows you to examine and modify each part of the HTTP transaction, beginning with the client's initial contact through the logging of the transaction on the server's file system. Each HTTP server divides an HTTP transaction into a series of stages; Apache has more than a dozen such stages.

Each stage is known as a “handler” and is given the opportunity to act on the current stage of the HTTP transaction. For example, the TransHandler translates URLs into files on the file system, a LogHandler takes care of logging events to the access and error logs, and a PerlTypeHandler checks and returns the MIME type associated with each document. Additional handlers are called when important events, such as startup, shutdown and restart occur.

Each of these Apache handlers has a mod_perl counterpart, known by the collective name of “Perl*Handlers”. As you can guess from this nickname, each Perl*Handler begins with the word “Perl” and ends with the word “Handler”.

A generic Perl*Handler, known simply as PerlHandler, is also available and is quite similar to CGI programs. If you want to receive a request, perform some calculations and return a result, use PerlHandler. Indeed, most applications that are visible to the end user can be done with PerlHandler. The other Perl*Handlers are more appropriate for changing Apache's behavior from a Perl module, such as when you want to add a new type of access log, alter the authorization mechanism, or add some code at startup or shutdown.

I realize the distinction between Perl*Handlers (meaning all of the possible handlers available to Perl programmers) and PerlHandlers (meaning modules that take advantage of Apache's generic “handler”) can be confusing. Truth be told, confusing the two isn't that big a deal, since the majority of programs are written for PerlHandler and not for any of the other Perl*Handlers.

As I mentioned above, mod_perl caches Perl code, compiles it once, then runs that compiled code during subsequent invocations. This means that, in contrast to CGI programs, changes made in our program will not be reflected immediately on the server. Rather, we must tell Apache to reload our program in some way. The easiest way to do this is to send a HUP signal (**killall -1 -v httpd** on my Linux box), but there are other ways as well. Another method is to use the **Apache::StatINC** module, which keeps track of modules' modification dates, loading new versions as necessary.

Writing a Simple PerlHandler

As we know, CGI programs are stand-alone programs that are invoked from an outside process, namely the web server. PerlHandler modules are actually subroutines within the Apache process; Apache invokes our subroutine when a certain set of conditions is fulfilled.

Writing a PerlHandler module is not much different from writing any Perl module. (If you are unfamiliar with writing Perl modules, see the “perlmod” man pages, or any of the books available on the subject.) We create a module with a single subroutine defined, called “handler”, shown in Listing 1. This code has several elements common to many PerlHandler modules.

Listing 1

Listing 1. PerlHandler Module

First of all, the entire module contains a single subroutine, “handler”. We can define additional subroutines if we want, but usually it is easiest to use the established standard and default.

Next, notice the handler is invoked with a single argument, which we call **$r.** It is an instance of the Apache object, which gives us access to the innards of the Apache web server. **$r** is our conduit to the outside world of the HTTP server and the user's browser. We invoke certain methods to determine the state of the server and browser and other methods to send output to the user's browser. Without **$r** we are somewhat lost, so it is natural that our first action upon entering “handler” is to retrieve **$r**.

We also use the **-w** and **use strict** programming aides in our program. While these are normally good ideas for good, clean Perl programs, they are essential when developing under mod_perl. As we will see later, mod_perl's caching and persistence means we need to be extra careful with our use of memory, in order to keep our HTTP server process as slim as possible.

Our handler uses only three methods from **$r: content_type**, **send_http_header** and **print**.

The first method, content_type, allows us to set or retrieve the “Content-type” header that will precede the response. Every HTTP response must be described with such a header, which tells the browser whether the response is an HTML-formatted text file, a GIF image or a **zip** file.

Once we have set the “Content-type” header to an appropriate value, we send all of the headers to the user's browser with the send_http_header method. Past this point, anything sent to the user's browser will be considered part of the HTTP response body, rather than the headers that describe that body.

The third method, print, is analogous to the built-in “print” function. However, it takes into consideration several factors that “print” might not, such as timeouts. **$r->print** takes a list of arguments just as the “print” function does. Thus, you can use

```
$r->print("a", "b", "c");
```

and expect three characters to be sent to the user's browser.

Constants and Return Codes

Once we have finished writing the response, we exit from our module by returning the OK symbol to the caller. We import OK from **Apache::Constants**, a module that provides us with a large number of useful symbols. In order not to pollute our name space too much, we explicitly request that only “OK” be imported with no other symbols.

If we were writing a more complicated module, we might use one of the export tags such as **:common** and **:response**, which allow us to import a group of symbols without having to name them explicitly. Thus, we could use the statement:

```
use Apache::Constants qw(:response);
```

which would import all symbols needed for a response.

Most PerlHandler modules will want their “handler” subroutines to return one of two symbols: either OK, which indicates that the handler successfully dealt with the request and no other PerlHandler needs to do anything, or the DECLINED symbol. If your module's “handler” routine returns DECLINED, it means “I was unable to do anything with the input I was given and would be happy if some other PerlHandler would do something.” Often, returning DECLINED means the default Apache behavior will be applied; if our PerlHandler were to return DECLINED, Apache would try to read the file named in the URL and do something with it. By returning OK, we indicate that our module took care of things, and Apache can move on to the next PerlHandler.

Configuring Apache

Now that we have seen how easy it is to write a PerlHandler module, let's look at how to install this module on our web server. We do this in the configuration file, typically named httpd.conf. If your copy of Apache uses three .conf files, understand that the division between them is artificial and based on the server's history, rather than any real need for three files. Apache developers recognized this increasingly artificial division and recently decided that future versions of the server will have a single file, httpd.conf, rather than three.

Apache configuration files depend on directives, which are variable assignments in disguise. That is, the statement

```
ServerName lerner.co.il
```

sets the “ServerName” variable to the value “lerner.co.il”.

If you want a directive to affect a subset of the files or directories on the server, you can use a “section”. For instance, if we say:

```
<Directory /usr/local/apache/share/cgi-bin>
AllowOverride None
Options ExecCGI
</Directory>
```

then the **AllowOverride** and **Options** directives apply only to the directory /usr/local/apache/share/cgi-bin. In this way, we can apply different directives to different files.

“Directory” sections allow us to modify the behavior of particular files and directories. We can also use “Location” sections to modify the behavior of URLs not connected to directories. Location sections work in the same way as Directory sections, except that Location takes its argument relative to URLs, while Directory takes its argument relative to the server's file system.

For example, we could rewrite the above Directory section as the following Location section:

```
<Location /cgi-bin>
AllowOverride None
Options ExecCGI
</Location>
```

Of course, this assumes that URLs beginning with /cgi-bin point to /usr/local/apache/share/cgi-bin on the server file system.

Installing a PerlHandler Module

All this background is necessary to understand how we will install our PerlHandler module. After all, our PerlHandler will influence the way in which one or more URLs will be affected. If we (unwisely) want our PerlHandler module to affect all the files in /cgi-bin, then we use

```
<Location /cgi-bin>
SetHandler perl-script
PerlHandler Apache::TestModule
</Location>
```

This tells Apache we will be handling all URLs under /cgi-bin with a Perl handler. We then tell Apache which PerlHandler to use, naming **Apache::TestModule**. If we did not install Apache::TestModule in the appropriate place on the server file system and if the package was not named correctly, this will cause an error.

The above example is unwise for a number of reasons, including the fact that it masks all the CGI programs on our server. Let's try a slightly more useful Location section:

```
<Location /hello>
SetHandler perl-script
PerlHandler Apache::TestModule
</Location>
```

The above Location section means that every time someone requests the URL "/hello" from our server, Apache will run the "handler" routine in Apache::TestModule. Because we used a Location section, we need not worry whether /hello corresponds to a directory on our server's file system.

This is how mod_perl creates a status monitor:

```
<Location /perl-status>
SetHandler perl-script
PerlHandler Apache::Status
</Location>
```

Each time someone requests the /perl-status URL from our server, the **Apache::Status** module is invoked. This module, which comes with mod_perl, provides us with status information about our mod_perl subsystem. Again, because we use a Location section, we need not worry about whether /perl-status corresponds to a directory on disk. In this way, we can create applications that exist independent of the file system.

Once we have created this Location section in httpd.conf, we must restart Apache. We can send it an HUP signal with

```
killall -HUP -v httpd
```

or we can even restart Apache altogether, with the program **apachectl** that comes with modern versions of the server:

```
apachectl restart
```

Either way, our PerlHandler should be active once Apache restarts.

We can test to see if things work by going to the URL /hello. On my home machine, I pointed my browser to http://localhost/hello and received the "testing" message soon after. If you don't see this message, check the Apache error log on your system. If there was a syntax error in the module, you will need to modify the module and restart the server as described above.

The first time you invoke a PerlHandler module, it may take some time for Apache to respond. This is because the first time a PerlHandler is invoked on a given Apache process, the Perl system must be invoked and the module

loaded. You can avoid this problem to a certain degree with the **PerlModule** directive, described later in this article.

Results

The subroutine we just created might seem trivial, but it demonstrates the fact that we can easily modify the behavior of our web server simply by writing a Perl subroutine. Moreover, since subroutines can contain just about any sort of Perl code, we have at our disposal all of the Perl modules, operators, functions and regular expressions that would be available to a stand-alone program.

Indeed, our "handler" routine is simply an entry point to what can be a large, complex program with other subroutines. Since Perl*Handler modules have access to Apache at every stage of operation, we can modify anything using Perl. A growing library of modules that do many common tasks is available, so that you can spend time on the particulars of your problem, rather than reinventing the wheel.

Another Module

Let's write another PerlHandler module, but this time let's have it do something other than return its own output. Just for fun, we will have it turn headlines in a file into Pig Latin. (In Pig Latin, the first letter of each word is moved to the end of the word, and "ay" is tacked on to the end.)

We will call our PerlHandler module **Apache::PigLatin**, which means we will create a module named PigLatin.pm and put it into the Apache module subdirectory. The source code is shown in Listing 2.

Listing 2

We install our module with a Directory section in httpd.conf:

```
<Directory /usr/local/apache/share/htdocs/stuff>
SetHandler perl-script
PerlHandler Apache::PigLatin
</Directory>
```

Make sure the directive points to an actual directory in your Apache document tree.

The module introduces several new ideas, but nothing revolutionary. For starters, we import the constants OK, DECLINED and NOT_FOUND. As we indicated earlier, we will use OK to indicate that our PerlHandler did something, and DECLINED to indicate that Apache should apply some other behavior. We will use DECLINED to ensure our PerlHandler works on HTML-formatted text by checking **$r->content_type**. If the MIME type is "text/html", we will operate on

the file. If it is a JPEG image, we will refrain from translating it into Pig Latin, returning DECLINED.

Next, we attempt to open the file from **$r->filename**. This particular module is being used as a simple PerlHandler, so we can be sure the translation from URL to a file name on the file system has been performed. This translation takes place in the TransHandler stage, which we can modify by writing a PerlTransHandler, rather than a simple PerlHandler. While it has translated the URL into a file name on our system, Apache has not checked to see if the file exists—that is our job. If we cannot open the file, we will assume it does not exist, returning the symbol NOT_FOUND.

Now things get interesting: we grab the contents of the file and perform a substitution on headlines—that is, anything between **<H\d>** and **</H\d>**, where **\d** is a built-in character class matching any digit.

We use **.*?** to match all characters rather than a simple **.***, so as to turn off the "greedy" feature in Perl's regular expressions. If we were to say **.*** rather than **.*?**, we would match all characters between the first <H\d> and the final </H\d>, rather than between the first pair, the second pair, and so forth. Greediness is usually a good thing when working with regular expressions, but can be frustrating under these circumstances.

We use four options in our substitution, using evaluation (**/e**), case-insensitivity (**/i**), global operation (**/g**) and the **.** regexp character to match **\n** (**/s**). This allows us to perform the substitution in one fell swoop, as well as catch any headlines that might begin on one line and continue on the next one.

Inside the substitution we invoke **pl_sent**, which is a subroutine defined within our module. This subroutine is not invoked directly from mod_perl, but is there to assist our "handler" routine in doing its work.

What's more, pl_sent invokes another subroutine, **piglatin_word**, which translates words into Pig Latin. If we were interested in creating a large web application based on mod_perl, you can see how it would be possible to do so, creating a number of subroutines and accessing them from within "handler". C programmers might think of "handler" as the mod_perl equivalent of "main", the subroutine invoked by default. Once in that routine, you can do just about anything you wish.

The pl_sent routine is interesting if you have never stacked **split**, **map** and **join** before. We split **$sentence** into its constituent words across **\s+**, which represents one or more whitespace characters. We then operate on each element of the resulting list with map, running piglatin_word on each word.

Finally, we piece together the sentence in the end, using join to add a single space between each word. The result is returned to the calling **s///** operator, which inserts the translated text in between the headline tags.

It is a much tougher problem to handle paragraphs, partly because people often forget to surround paragraphs with <P> and </P>, relying on the fact that browsers will forgive them if they simply say <P>. In addition, paragraphs contain punctuation which makes a good Pig Latin translator harder to write.

There is no limit to the kind of filters you can write. Perhaps the most interesting and advanced are those that use Perl's **eval** operator to evaluate little pieces of Perl code inside HTML files. A number of these already exist, such as Embperl (discussed several months ago) and EPerl. More simply, you can ensure that every file on your system has a uniform header and footer, removing the need for server-side includes at the top and bottom of each file.

Reducing Memory Usage

mod_perl is an exciting development that has already made a great many new applications possible. But there is a trade-off for everything, and mod_perl's additional functionality comes at the expense of greater memory usage. It is hard to calculate the additional memory needed for mod_perl, but keep in mind that Perl can be a bit of a memory hog.

In addition, while lexical (“my” or “temporary”) variables disappear after each invocation of a Perl module rule via mod_perl, global variables stick around across invocations. This can be an attractive way to keep track of state in your program, but it can also lead to larger memory allocations.

For example, if your module creates an array with 10,000 elements, that array will continue to consume memory even after the program is invoked. This might be useful in some cases, such as when a complicated data structure is referenced in each invocation. However, it also means the large structure will constantly eat up memory, as opposed to only when necessary.

You can reduce memory usage by forcing mod_perl to share memory among Apache child processes. When you run Apache as a web server, it “preforks” a number of processes so that incoming connections will not have to wait for a new server process to be created. Each of these preforked servers is considered a separate process by Linux, operating independently. However, Apache is smart enough to share some memory among server siblings, at least to a certain degree.

mod_perl takes advantage of this shared memory by allowing the various server processes to share Perl code as well. However, there is a catch: you must

make sure the Perl code is brought into mod_perl before preforking takes place. Perl modules and code compiled after the split occurs will raise the memory requirement for each individual server process, without regard to whether the same code has been loaded by another process.

In order to load code before Apache forks off child processes, use the PerlModule directive in the configuration files.

If, for example, you use the statement

```
PerlModule Apache::DBI
```

in one of the *.conf files, then

```
use Apache::DBI;
```

in a PerlHandler module, the latter invocation does not actually load any new code. Rather, it uses the cached, shared version of **Apache::DBI** that was loaded at startup by mod_perl.

You can load multiple modules with PerlModule, using the syntax

```
PerlModule Apache::DBI Apache::DBII Apache::DBIII
```

However, you can load only ten modules this way. If you want to load more, you can use the PerlRequire directive. Strictly speaking, PerlRequire allows you to specify the name of a Perl program to be evaluated only when Apache starts up. For example,

```
PerlRequire /usr/local/apache/conf/startup.pl
```

will evaluate the contents of startup.pl before forking off Apache child processes. However, if you include a number of “use” statements in startup.pl, you can effectively get around PerlModule's ten-module limit.

Remember that PerlModule or PerlRequire is necessary for modules to be shared among the different Apache sibling server processes, but it is not sufficient. You will still have to import the module in your own program in order to reap the benefits.

Conclusion

When I first started to work with mod_perl, I thought it was useful for speeding up CGI programs and for running filters like Embperl. As I have grown more dependent on it in my own work, I am amazed and impressed by the power

mod_perl offers programmers looking to harness the power of Apache without the overhead of external programs or the development time associated with C.

As you can see, writing mod_perl modules is not difficult and is limited only by your imagination. It does require that you think a bit more carefully about your programs than when you are working with CGI, since you can affect the Apache server in ways that will slow it down or otherwise hurt your system's performance.

Resources

Reuven M. Lerner (reuven@lerner.co.il) is an Internet and web consultant living in Haifa, Israel, who has been using the web since early 1993. His book Core Perl will be published by Prentice-Hall in the spring. The ATF home page, including archives and discussion forums, is at http://www.lerner.co.il/atf/.

Archive Index Issue Table of Contents

Advanced search

Security Research Laboratory and Education Center

Sofie Nystrom

Issue #60, April 1999

The world-class research center at Purdue University is getting serious about cutting edge development of security related projects.

Keeping the bandits out is not the only reason you will need educated security experts to maintain your system in the future. What will happen when the demand for security administrators is so high your firm cannot afford them? Or if the total development cost for secure software is more than the debt in the U.S. alone? Your answer to my last question may be, “We will use open-source code”--good point! However, you will still need experienced security personnel to maintain your system.

As most of the industry is struggling to prepare their systems for the year 2000, academia is facing the problem of educating enough computer scientists. Government reports predict that in the year 2000, on-line commerce in the U.S. alone will exceed 15 billion dollars per year, and the sales of security software will exceed two billion dollars per year. The need for increased training and research in information security will only expand in the coming years as the use of wide-area computer networks spreads.

Fighting Information Warfare with Education

As computer crime is increasing, Purdue University in Indiana is addressing the issue. For the last seven years, the Purdue Computer Science Department has been the home of the Computer Operations, Audit and Security Technology (COAST) laboratory. COAST is one of the largest academic research groups and graduate studies laboratories in practical computer and network security in the world. The laboratory is expanding into a newly established center.

Purdue's University Center for Education and Research in Information Assurance and Security (CERIAS as in “serious”) is a pioneer in the area of information security. This new university center was designed to educate the

next generation of computer and network security specialists. With projects encompassing Linux, Solaris, Windows 95/NT, smart cards, iButtons, biometrics, ATM networks and firewalls, their research will work toward the goal of reducing the threat of so-called information warfare.

One of a Kind

The director of the laboratory and of the newly founded center, Professor Gene Spafford, is a computer scientist who has been a major contributor to the discipline of information security. Spafford is an ACM (Association for Computing Machinery) fellow and has written several books on information security. He also helped to analyze and contain the Internet worm in 1998. Together with 15 faculty members and 40 graduate and undergraduate students (see Figure 1), he is steering the center toward a common goal: to provide world-class research and education in information security.

Figure 1. Professors, staff and students in the COAST/CERIAS facility at Purdue University

Currently, the faculty and students are drawn heavily from the computer science area. However, the center is opening its doors to a diversity of disciplines (e.g., philosophy, linguistics, political science, industrial engineering, management, sociology and electrical and computer engineering).

Figure 2. The Graduate Student COAST Laboratory

The laboratory (see Figure 2) and the new center have attracted professors and students from 13 countries. One reason is that there are few highly competent academic security laboratories with industry support. The diversity does not end with nationality—almost 40 percent of the students are female. Security has drawn the interest of women since the early days, and the number of female students has been increasing steadily in the last few years.

The research includes audit trails format and reduction, network protection, firewall and software evaluation, creation of a vulnerabilities database and testing. Additionally, several undergraduate projects dealing with authentication and security archive are in progress. The main COAST projects are described briefly below.

A Different Approach to Intrusion Detection

Intrusion Detection (ID) is a field within computer security that has grown rapidly over the last few years. The AAFID (autonomous agents for intrusion detection) project in the COAST laboratory is about intrusion detection.

Traditional intrusion detection systems (IDS) collect data from one or more hosts and process the data in a central machine to detect anomalous behavior. This approach has a problem in that it prevents scaling of the IDS to a large number of machines, due to the storage and processing limitations of the host that performs the analysis.

The AAFID architecture uses many independent entities, called “autonomous agents”, working simultaneously to perform distributed intrusion detection. Each agent monitors certain aspects of a system and reports strange behavior or occurrences of specific events. For example, one agent may look for bad permissions on system files, another agent may look for bad configurations of an FTP server, and yet another may look for attempts to perform attacks by corrupting the ARP (address resolution protocol) cache of the machine.

The results produced by the agents are collected on a per-machine level, permitting the correlation of events reported by different agents that may be caused by the same attack. Furthermore, reports produced by each machine are aggregated at a higher (per-network) level, allowing the system to detect attacks involving multiple machines.

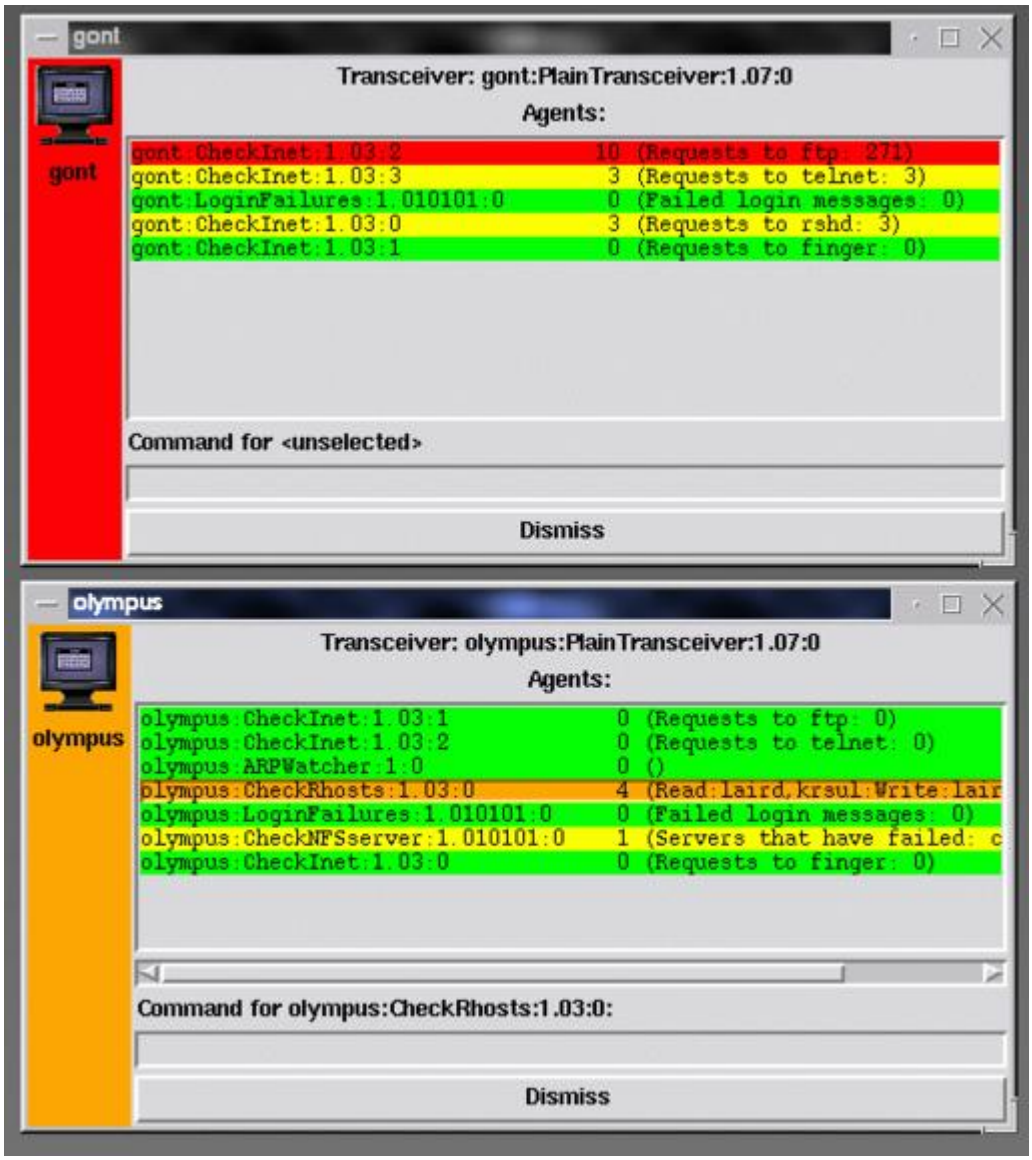


Figure 3. The Agent Window of the AAFID Prototype

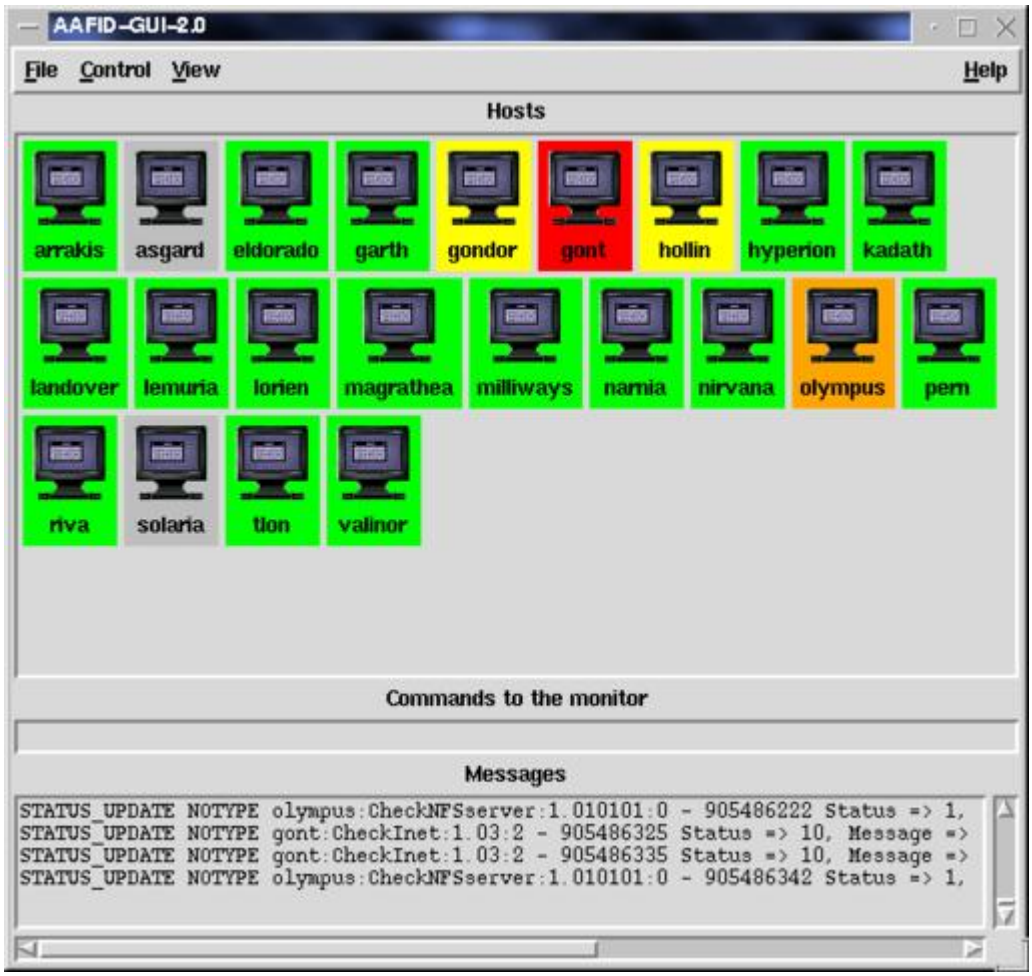


Figure 4. The Main Window of the AAFID Prototype

The AAFID group consists of ten graduate and undergraduate students within the COAST laboratory. A prototype implementation (see Figures 3 and 4) can be found on the AAFID project web page at http://www.cs.purdue.edu/coast/projects/autonomous-agents.html.

Security Archive

During the past several years, students at Purdue have been maintaining the Internet's largest on-line archive of security-related tools, papers, standards, advisories and other materials. The main problem they face is the efficient management of such highly dynamic information. The group will mirror constantly changing sites and will maintain the most recent copies of those sites. Additionally, new sites are continually starting up—new papers, new tools and more information that must be added to the archive.

The other major concern is being able to find what they are looking for in the archive. With so much information, it is difficult to navigate through all the data. The Archive group has used a combination of Red Hat Linux 5.2 and the open source ROADS (see Resources) document ordering system to build the prototype. This will transform the FTP-based archive to an HTTP-based

information system, allowing users to search based on different criteria or enter a Yahoo-like browsing mode. The group always welcomes contributions and suggestions (security-archives@cs.purdue.edu).

Tripwire

One of COAST's better-known projects is Tripwire. It was primarily a project of Gene Kim and Professor Gene Spafford. The product is now used worldwide and is the most widely deployed intrusion detection security tool. Tripwire is an integrity monitoring tool for Linux and other UNIX systems. It uses message digest algorithms to detect tampering with file contents that might have been caused by an intruder or virus. In December 1997, Visual Computing Corporation obtained an exclusive license from Purdue University to develop and market new versions of the product. Tripwire IDS 1.3 has been released for Linux. For more information, see the web site http://www.tripwiresecurity.com/.

Underfire

The Underfire team consists of seven graduate and undergraduate students. Their goals are to gain direct experience in the installation, evaluation, configuration and usage of different firewall systems; to investigate new technologies for network perimeter defenses, including next-generation networks such as ATM; and to investigate the integration of host- and network-based security mechanisms with network perimeter defenses. Underfire is an ongoing project which began in 1997.

The Underfire team's main goal is to create an architecture for automated firewall testing. The final product will be an engine that will test a firewall without human interaction. This will be achieved with a modular system: the engine, a packet sniffer and scripted attacks. The engine executes the attacks and uses the packet sniffer, or other networking protocols, to test the success or failure of the attack. Finally, a report can be automatically generated that explains the weak points of the firewall based on the attack data.

Having finished the design and initial implementation of the engine, the Underfire team is currently scripting known attacks. The automatic report generator is something that will also need to be completed in the future. Until now, Underfire has taken only protocol level attacks into account; a future step will be to extend the tests to the application level such as RPC and X11.

Next Generation Authentication

The need to change the old-fashioned login name and password procedure of authentication is an obvious place to base research for the laboratory. By using

biometrics devices and tokens such as smart cards and iButtons, several research and application development projects can be conducted in this area. Security tokens under Linux will provide a wide array of security features for the multiuser operating system.

One of the COAST students heads the MUSCLE project (Movement for the Use of Smart Cards in a Linux Environment), one way to integrate security tokens into the Linux environment. MUSCLE focuses on smart card and biometrics security under Linux and consists of several projects. The first is standardizing on a PC/SC-compliant smart card resource manager written in C++, along with cryptographic libraries based on the Public-Key Cryptography Standards (PKCS-11 and PKCS-15). The resource manager also allows secure remote authentication by using secure channels to communicate between multiple resource managers. The resource manager will be used to develop many applications, including secure login, ssh, xlock, FTP, TELNET, et al., via pluggable authentication modules (PAM) along with smart card security.

MUSCLE supports a wide array of smart card readers along with ISO-7816-4-compliant smart cards. On the web site, you can find many different smart card specifications, source code for different projects, on-line tutorials and a mailing list. MUSCLE can be found at http://www.linuxnet.com/.

Enhancing the Linux Audit Trail

COAST graduate students have been studying ways of enhancing audit trails on Linux systems. Additionally, penetration and vulnerability analysis efforts have benefited from the use of Linux machines with the enhanced auditing systems.

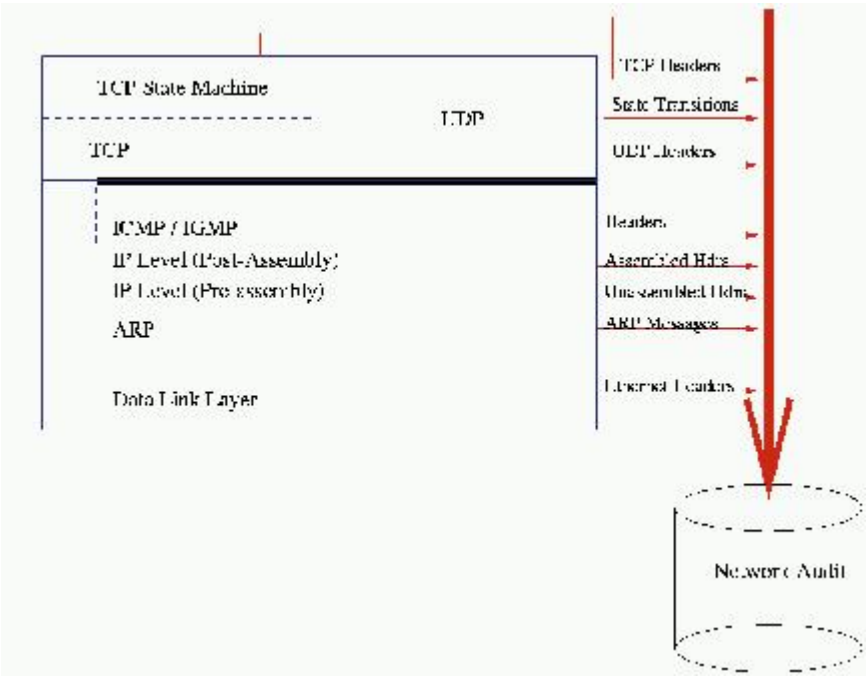


Figure 5. Network Auditing Trail

Generally, operating systems' audit trails or logs are inadequate for a variety of applications such as intrusion detection. The students have developed two different approaches to enhancing the data collected by Linux. One approach was to use the technique of interposing shared objects to collect new application-level audit data. Using this technique, a program can be instructed to record and act upon certain library calls and their arguments without modifying the binary or source code of the program. (See Figure 6.)

Figure 6. Snapshot of the Kernel Code

Another part of the project involves using a Linux 2.0.34 kernel (see Figure 7.) to audit low-level network data. This involves adding a mechanism to the kernel to report network packet headers to user processes. By correlating these data and intrusion detection systems, host-based intrusion detection systems can detect low-level network attacks such as “Land”, “Teardrop” and “Syn floods”. This

mechanism uses a version of the existing kernel log code, modified to accommodate arbitrary binary data.

Vulnerabilities Database and Testing

The vulnerability database and analysis group at COAST is collecting and analyzing computer vulnerabilities for a variety of purposes. The project includes the application of knowledge discovery and data mining tools to find non-obvious relationships in vulnerability data, to develop vulnerability classifications and to develop tools that will generate intrusion detection signatures from vulnerability information. One goal of the group is to develop methods of testing software in order to discover security flaws before the software is deployed.

Future

In the words of Professor Spafford:

> With the increasing use of computers and networks, the importance of information security and assurance is also going to increase. Concerns for privacy, safety and integrity may soon become more important to people than speed of computation. This represents a tremendous challenge, but also a tremendous opportunity for those who seek to understand—and provide—workable security.

Resources

Sofie Nystrom is a Computer Science undergraduate student at Purdue University, where she is doing research for COAST laboratory and developing applications for the use of smart cards and biometrics for Linux. She is also working for CERIAS and is involved in the Computer Science Women's Network. She can be reached at sof@cs.purdue.edu.

Archive Index Issue Table of Contents

Advanced search

Windows/Linux Dual Boot

Vince Veselosky

Issue #60, April 1999

Don't want to give up Windows while you learn Linux? Here's how to use both on the same machine.

So you've heard great things about Linux: faster, cheaper, more efficient, more stable. Sounds good. You'd like to try it out, but probably have a lot of time and data invested in Windows and can't afford to be down while figuring out how to use Linux for your daily tasks. Windows and Linux can live comfortably on the same computer, even on the same hard drive. The choice of operating system can be made when you power on. This is commonly called a “dual boot” configuration, and one of the most common questions among new Linux users is how to set it up.

My system is a Pentium II 400MHz with 128MB of RAM and an 11GB EIDE (actually Ultra-DMA 33 ATAPI, for you hardware gurus) hard drive. The hard drive had Windows 95 “C” on one big FAT32-formatted C: drive, which is a typical factory configuration. I tested installs of Red Hat Linux 5.1 and SuSE Linux 5.2.

Before starting, there are two terms you need to be familiar with: *partition* and *file system*. The disk can be divided into smaller, separate pieces which can belong to different owners. For dual booting, Windows will own some and Linux will own others. The word “partition” does not refer to the wall; it refers to the separated space. Thus, we say Windows is installed “on” the first partition. The *file system* is a method of organization. Your hard drive can have different file systems. The operating system provides the *directory tree* (also referred to as “the file system”) as a catalog of available files. Every operating system has its own type of file system, and other operating systems often don't know how to read it. Lucky for us, Linux is a versatile operating system and it does understand the file system used by Windows 95 and Windows 98.

Making Room for Linux

Most factory-installed Windows installations take up all the space on your hard drive, leaving no room for installing Linux. The first and most difficult thing we must do is clear some space where Linux can be installed. Linux needs to have partitions of its own, but Windows does not have the ability to resize partitions. Ordinarily, this would mean you would have to delete your existing partition (and everything on it) to make room on the drive and then create partitions of smaller sizes and reinstall. You can still do this, but there is a better way.

What You Need

Most Linux distributions come with a special tool to allow you to resize or divide hard drive partitions. Called FIPS, the First (non-destructive) Interactive Partition Splitter, it is normally found on your Linux CD in a directory called /dosutils. You will also need a blank, formatted floppy disk to use as a boot disk. For most older Windows installations, that should be all. However, if you have Windows 98 or a recent version of Windows 95 with a large hard drive (bigger than 2GB), you may need some additional tools if you are using the FAT32 file system.

To check what type of file system Windows is using, open Windows Explorer, right click on the C: drive and choose Properties. If you see “File System: FAT32” on the General tab, you will need some additional tools to support this newer file system.

FAT32 Support Requirements

To adjust your partitions, you will need version 2.0 or higher of FIPS. If the version included with your Linux distribution is older than this, the latest version is available for download from the FIPS home page at http://www.igd.fhg.de/~aschaefe/fips/. If you want to share files between Windows and Linux (a good idea), you will also need to have version 2.0.34 or higher of the Linux kernel. Table 1 is a list of Linux distributions known to support FAT32. If your distribution does not include support, you will need to upgrade the kernel. Upgrading a kernel is beyond the scope of this article, so check the documentation included with your distribution or your distributor's web site for information about how to do that.

Table 1

Preparing Your Drive

Before you can resize your Windows partition, a few steps must be taken to ensure that the process goes smoothly. First, delete any files from your hard

drive that are not being used; for example, any old files in the C:\windows emp folder, and then empty your recycle bin. Next, check your file system for errors using Scandisk, and compact your hard drive using Defrag. I'll assume you Windows users know how to do this. When running Scandisk, be sure to check the box next to “Automatically Fix Errors”. Defragmentation consolidates all your data at the “front” of the drive to make room at the “back” of the drive for your new partition.

When both are finished, it would be wise to note how much space is available on the disk. If this number is less than the amount required to load Linux (check your distribution's documentation for space requirements), you'll need to delete more files or uninstall some software to make room.

Using FIPS

Before using FIPS, you *must* read the FIPS.DOC text file which accompanies the program. The use of the program is not entirely obvious, and you may need the background information the documentation provides. Also, while running FIPS you should carefully read all the messages it displays. They will provide valuable information on the steps you will need to take next. Most importantly, FIPS comes with no warranty. Although it has been used safely many times, there is always the chance it could damage the data on your hard drive. If you value your data, *back it up* before you begin.

For safety, create a DOS or Windows boot disk to work from. To do this, click Start **->** Settings **->** Control Panel. Double click “Add/Remove Programs” and select the tab called Startup Disk. Press the button and follow the instructions.

Next, copy the working files for FIPS to the floppy. The files FIPS.EXE, RESTORRB.EXE and ERRORS.TXT are mandatory. You may also want to copy the documentation files included with FIPS. When your data is backed up, restart your computer and boot from the new floppy.

When you arrive at the **A:>** prompt, type **FIPS** and press enter. A warning will appear about using FIPS in multitasking environments like Windows. Since we booted from a floppy, we are safe, so press enter. FIPS will analyze your existing partitions. It may pause for a long time at “Checking FAT” and “Searching for Free Space”; this is perfectly normal, so just wait. The bigger your hard drive, the longer it will take. When FIPS is done with its analysis, it will display the results. You may get a warning of something being wrong with your FAT. If you read the message carefully, you will find that this is normal with large hard drives and will not prevent FIPS from working properly.

FIPS will then demonstrate how it plans to split the existing partition and you will have the opportunity to make changes. Do *not* just press enter. By default, FIPS will take *all* of the available free space for the new partition it creates, leaving your Windows partition with no free space at all. Windows will not run if it has no free drive space, so you *must* adjust the partitions. Use the up and down arrow keys to make large changes (ten cylinders at a time) and the left and right arrow keys for small adjustments (one cylinder at a time). The size of the existing partition is shown on the left and the size of your new empty partition is on the right. In the middle is the cylinder number where the split will take place. I left about 1500MB for my own Windows partition. Adjust yours according to your needs, but I would recommend using at least 1024MB for Windows.

When you are satisfied, press enter. FIPS displays information on the new partitions and asks permission to write it to disk. Your hard drive has not been altered at this point. You may choose to write this configuration to disk or re-edit the partition table. On my machine, when I chose to re-edit I received an error message that said FIPS couldn't find some files it needed. If this happens to you, just press **ctrl-alt-delete** to reboot from the floppy and start over. This did not cause me any trouble.

When you choose to write the new partitions, FIPS will offer to make a backup of your existing boot sector—you should definitely do this. The backup file it creates is only 1KB in size and will be invaluable if anything goes wrong.

After FIPS completes its work, it will display another message stating that you should run scandisk on your old partition. I found that Windows will sometimes miscalculate the used and free space on your drive after using FIPS, and Scandisk will correct this problem. If you choose to restore your original partition scheme using the RESTORRB utility, you should run Scandisk after this as well.

After FIPS was done, I received another error. This one said “Memory Allocation Error, Unable to Load COMMAND.COM”. If you see this, just press **ctrl-alt-delete** to reboot and all is well. This should not affect your hard drive.

Finally, you may want to run the Windows FDISK program from your floppy. This is not necessary, since Linux has its own fdisk program for manipulating partitions. You will find that your hard drive now contains two “Primary Partitions” (or “Primary DOS Partitions”). The second one was created by FIPS out of the free space on your drive. For Linux installation, delete this second partition, freeing up the space for allocating Linux partitions. (Be careful not to delete the first one, where Windows lives.)

Linux Install Tips for Large Drives

Once you've made room for Linux on your drive with FIPS, you should be able to install Linux by following the steps in the installation guide that accompanied your Linux distribution. Here are a few tips that should help you with the areas where dual booting might make a difference.

Planning your Partitions

Both the Red Hat and SuSE installation guides have excellent chapters on how to divide up your hard drive for use by Linux. Personally, I favor the “Keep It Simple” principle, especially for beginners. I let Windows keep the first partition, create a second for the entire Linux install, a third for Linux swap space and the fourth for my /home directory (where data is kept). Having /home on a separate partition will make things much easier, if you ever have to reinstall Linux. The size of each partition will depend on your individual situation, but this should suffice for most folks. However, if your hard drive is larger than 8GB, there is something else to think about—LILO.

Booting with LILO

The usual and recommended method to boot into Linux is using LILO (the LInux LOader). LILO can install itself in your boot sector and allows you to choose which operating system you would like at boot time. Due to a technical limitation, LILO is unable to read data from the hard drive past the 1024th cylinder—the 8GB mark for modern LBA (Logical Block Addressing) hard drives.

Does this mean you can't use the rest of your drive? Not at all. What it does mean is that your *boot partitions* must all live *below* the 8GB mark, that is, below cylinder 1024. Thus, if you want Windows to use the first 9GB of your fancy new 18GB drive, you won't be able to use LILO to boot Linux. Because of this limitation, Red Hat's Disk Druid tool for partitioning the hard drive will not allow you to create your Linux boot partition past cylinder 1024. You can still create the partitions using fdisk, but Red Hat setup will not install LILO if you do.

Booting from Floppy

It is possible to avoid the entire problem of the 8GB barrier by booting from a floppy disk. Although this may sound inefficient, it actually works quite well. The kernel loads into memory from the floppy disk and never accesses the floppy again, so loading the kernel is slower; but after that, the system runs the same as if it had booted from the hard drive. The Linux kernel has no difficulty accessing the end of large hard drives, so it can still reach all the files of your Linux installation.

The setup program for your distribution will almost certainly ask you to create a boot floppy during installation. Even if you don't plan to boot from floppy regularly, you should definitely make a boot disk. If for some reason LILO fails to install or becomes corrupted, you will have no other way to access the files on your Linux installation.

Booting with Loadlin

Loadlin is a program that runs under DOS (or Windows 95 in MSDOS mode). It can load the Linux kernel into memory from the DOS partition. Because it loads the Linux kernel from the hard drive, there is still a possibility the 8GB barrier could cause problems, but only if your Windows partition is larger than 8GB and is almost full. That's not likely at the time of this writing, but who knows—the next release of Windows might take up that much space by itself.

Frankly, I wouldn't recommend Loadlin to Linux novices because it can be difficult to configure correctly. If you simply must use it, an excellent Loadlin + Win95 Mini-HOWTO document available from the Linux Documentation Project should get you up and running.

Conclusion

Giving Linux a try does not mean you have to buy a whole new computer or even a new hard drive. With just a little extra effort, you can run both Linux and Windows without losing any data or any productivity while you learn Linux. I think you will find it is well worth the effort.

Resources

Vince Veselosky is a computer consultant in the Atlanta, Georgia area, working mostly in technical support for Microsoft operating systems. He has made it his mission in life to master Linux before the year 2000. When he's not working with computers, he's looking for a new girlfriend. Potential girlfriends and others can reach him via e-mail at vincevski@geocities.com.

Archive Index Issue Table of Contents

Advanced search

Focus on Software

David A. Bandel

Issue #60, April 1999

gtksamba, TkSmb, smb2www and more.

I am excited about Samba 2.0 and its new web interface called **swat**. However, there is still room for smaller, simpler tools that are found in the Samba package or do not require a web browser. We will look at a few of these tools today as well as a few graphical FTP packages.

gtksamba:

http://www.open-systems.com/gtksamba.html

gtksamba is a nice program that aids configuration and testing of the Samba smb.conf file. The smb.conf file to edit can be specified, if it has a different name or is located in a directory other than /etc. While not as detailed as swat, it allows a user with proper permissions to look at and edit smb.conf. This program appears just as powerful as swat, but does not require a web browser. Some features planned for this program are not yet implemented, such as the ability to configure remote machines. For its help, gtksamba uses the Samba man pages. However, it does not just present the man pages—it parses them to make finding specific parameters easy. In all, gtksamba is a well-thought-out program. It requires gtk 1.1.13, glib, Xext, X11, libm and glibc.

TkSmb:

http://www.rt.mipt.ru/frtk/ivan/TkSmb/

A Tcl/Tk interface to the **smbclient** program in Samba, **TkSmb** allows browsing of the “Network Neighborhood” in a box on your screen. Shown are the neighborhood hosts, the shares on any particular host, the different groups detected, and when a share is selected, the files in the share. Files and directories are displayed in black and blue respectively. One security aspect

requires you enter your password each time you change shares. A check box to "remember my password" for the current session would be helpful. It requires Tcl/Tk 8.0, expect 5.24+ and glibc 2.0.6.

smb2www:

http://us1.samba.org/samba/smb2www/

The **smb2www** package provides a view of the "Network Neighborhood" through a web browser and more closely resembles the Microsoft version. While nice, installation is a little difficult; the script walks you through questions, but could be a bit more friendly. Different hosts (Windows 9*x* or NT) show up as different icons—a nice touch. It requires a working web server (either Apache or another), Perl and a web browser.

LinPopUp:

http://www.littleigloo.org/

Under Windows, users have the ability to send each other messages if the messaging facility is turned on. The **LinPopUp** utility allows Linux Samba servers to exchange messages with Windows hosts or other Samba servers. It requires a change to the smb.conf file by root for messages to be received, but they can be sent without this modification. This facility will also receive messages from the NT server. It requires gtk 1.0.4+, Xext, X11, libm, Xmu, Xt, SM, ICE and glibc.

tkchooser2:

http://www.cs.columbia.edu/~etgold/software/tkchooser2/

tkchooser2 is another browsing tool that can be used with or without Samba. The default configuration is for AppleTalk (Netatalk) to be installed; however, this is easily changed. What this package lacks is a way to configure everything without opening several files and making changes. While the instructions are adequate, most newbies will not feel comfortable reconfiguring the package. Once a package is stable and ready for release, I would consider a configuration script that walks installers through initial setup a must. It requires Tcl/Tk 8.0.

gftp:

http://www.newwave.net/~masneyb/

gftp provides drag and drop for X. It is a well-done package that will help anyone move files around the Net. For doing file transfers, gftp is the easiest to use graphical utility I have seen, and it allows multiple transfers at one time (if

you have the bandwidth for it). Partial transfers can be resumed. Two windows below the two side-by-side directory windows allow you to watch transfers and messages from the host. An easy connection manager rounds out this utility. Utilities like this make it easy for newbies and give them one more reason to choose Linux. It requires libpthread, gtk 1.1, Xext, X11, lib and glibc.

IglooFTP:

http://www.littleigloo.org/

IglooFTP is another program from littleigloo. It is a graphical FTP package that is slightly larger than gftp. This one is still in the works, but already looks good. It is still alpha code, so is not yet as stable as gftp. One nice security feature is the ability to save a site, and when you connect, have the program request a password rather than storing it in the rc file. Also, your e-mail address (used as your anonymous password) is configurable from the preferences box. All in all, it is a very nice graphical FTP package. It requires gtk 1.1.11+, Xext, X11, libm, Xmu, Xt, SM, ICE and glibc.

xrmftp:

http://www.mat.uni.torun.pl/~rafmet/

xrmftp is yet another great FTP package. This one allows you to choose between active- and passive-mode transfers. Four windows include local and remote directories, a command viewer window and a buffer window. Files can be “queued up” in the buffer window, and when you have found the files you want, you can download the buffer. This permits you to look around without having to wait on a slow modem line while searching for more files. It requires xforms 0.88, X11 and glibc.

David A. Bandel (dbandel@ix.netcom.com) is a Computer Network Consultant specializing in Linux. When he's not working, he can be found hacking his own system or enjoying the view of Seattle from an airplane.

Archive Index Issue Table of Contents

Advanced search

Good Ol' sed

Hans de Vreught

Issue #60, April 1999

A nice little command to help you modify files.

When I started using UNIX in 1982, life in the computer world was fairly cruel. At that time, most programmers were still using line editors. The UNIX line editor, **ed**, was a relief in comparison with most line editors of other operating systems. Its sensible use of regular expressions was a blessing, and the fun part was that most UNIX tools used the same kind of regular expressions.

Although UNIX had virtual memory, the size of files that ed could handle was limited; disk space for memory was expensive. For large files, programmers had to resort to the stream editor, **sed**. **sed** reads its input line-after-line and performs its editing operations line-by-line. In sed, some commands allow use of multi-lines and so have a holding space, but in general, the amount of memory needed is small.

Besides the occasional one-line commands, I often wrote sed scripts. In those days, most system administration scripts were written in sed; **awk** was too slow and too big. The power demonstrated by those sed scripts was and still is quite amazing. They were true works of art—large and completely incomprehensible, but they got the job done.

Since sed is Turing-complete, it is as powerful as any programming language. Writing sed scripts that compute certain functions became a sport. Olaf Kirch, author of the *Linux Network Administrator's Guide*, says in his preface that he was proud to have written a prime number generator in sed. My pet script computes the Ackermann function and is available for anonymous download in the file ftp://ftp.linuxjournal.com/pub/lj/listings/issue60/2628.tgz along with a short explanation. It is just like programming in assembler.

Today, sed scripts are totally different—they are much simpler and are almost always one-liners. Most one-line sed commands (often included in Bourne

scripts or used interactively in your shell of choice) modify or delete certain lines in a file. In some cases, you might still write sed scripts; however, the commands remain simple. Beside the two operations just mentioned, you also insert, append and change groups of lines as a block.

The advanced sed commands have disappeared (and I must say I'm glad). Although these advanced commands made sed powerful, they also made the scripts unreadable. Today, if you need to do something advanced, you would use awk or Perl.

Syntax

I will not describe every feature of sed. Instead, I will restrict myself to just those commands I regularly use. For more information on sed, the best resource is *sed & awk* by Dale Dougherty and Arnold Robbins (O'Reilly & Associates, 1997).

sed commands have the following form with *no* trailing spaces:

```
[address][,address][!]command[arguments]
```

I will begin with the address. An address is either a line number ($ for the last line) or a regular expression enclosed in slashes. The regular expressions are similar to the ones you see in **vi** (well, actually the **ex** part of vi): “.” (any character), “*” (any number of the immediately preceding regular expression), “[class]” (any character in class), “[^class]” (any character not in class), “^” (begining of line), “$” (end of line) and "\" (to escape characters where needed).

A range of lines can be specified by giving two addresses. The “!” after the address specification excludes that range from being processed. The most commonly used sed commands are “d” (delete) and “s” (substitute). The delete command is straightforward; it deletes any line that matches the entire address specification. Substitute is more interesting:

```
s/pattern/replacement/[g]
```

Basically, *pattern* is just a regular expression, but it has an odd feature: parts of the pattern can be stored in the replacement. The parts to keep must be enclosed within the characters “\(” and “\)”. In the replacement part, these stored parts can be used by specifying “\1”, “\2”, ..., “\9” (the first, the second, ..., and the ninth stored part). If the entire matched part is to be used, the “&” character is specified. The **g** (global) flag can be used to replace all occurrences of *pattern* by *replacement*.

Example

Assume we have a small table of two columns of numbers which we wish to swap:

```
s/\([0-9]*\) \([0-9]*\)/\2 \1/
```

The "[0-9]" part matches any digit and the "*" means that we allow any string of digits to be matched. Since "[0-9]*" is surrounded by "\(" and "\)", it is stored. The first number of the line will be in "\1" and the second one will be in "\2". In the replacement, they are recalled: first the second number, then the first. While it is a bit silly to write a file with just two lines, it is:

```
#!/usr/bin/sed -f
s/\([0-9]*\) \([0-9]*\)/\2 \1/
```

The **-f** flag means that the next argument is a file name containing a sed script. The way UNIX executes scripts, this means the script name will be used as an argument of **-f** (i.e., this file). Here is the one-line equivalent to be used at the command-line prompt in the shell (the **-e** flag specifies that the next argument is a sed command):

```
sed -e 's/\([0-9]*\) \([0-9]*\)/\2 \1/'
```

To perform more than one command on a particular range, use { and } to bracket a block of commands. Assume that in the header of the file, we wish to replace C++ style comments ("// ...") with C style comments ("/* ... */") and we wish to update the copyright year. A blank line appears after the header, as shown here:

```
// Copyright 1997 John Doe
// This is just an example.
...
```

We can use the following script (note the heavy escaping by the backslashes) to modify this header:

```
#!/usr/bin/sed -f"
1,/^$/{
s/Copyright 1997/Copyright 1998/
s/\/\/\(.*\)/\/*\1 *\//
}
```

These commands will modify the above header to be:

```
/* Copyright 1998 John Doe */
/* This is just an example. */
...
```

Insert, Append and Change

The next set of sed commands I often use in scripts are "i" (insert), "a" (append) and "c" (change). Insert and append are similar in that they both need an address either before which or after which text is inserted or appended. The

text that is inserted follows on the next lines. All lines of the insert or append command must be “terminated” by a backslash except for the final line. For example:

```
1i\
This is a test\
to insert three lines\
at the beginning
```

With the change command, we can also specify a range of lines. If we specify only a single address, only that line will be changed; otherwise, the entire range will be modified.

Sometimes it is handy to read (“r”) from or write (“w”) to a file. In the case of a read, we can specify an address after which the file is to be read; in case of a write, we can specify one address or a range to be written to the file. For example, to write the lines preceding the first empty line to a file called foo, type the command:

```
#!/usr/bin/sed -f 1,/^$/w foo
```

sed is quite picky about spaces—only one space after “w” or “r” is allowed. Also, never use trailing spaces in sed commands; it doesn't like them and will give cryptic errors.

Next and Quit

The last two commands I will discuss are next (“n”) and quit (“q”). The next command takes a single address. After the specified line is output, the next line is read and the script resumes at the top. Normally, next commands are found within blocks of commands (i.e., within curly braces). The quit command also takes an address as an argument. When sed encounters quit, it immediately terminates the script without any further output.

Advanced

sed has several other commands, which can be considered advanced commands. You won't see them in modern sed scripts. Three sorts of advanced commands are available:

- Conditional commands: in sed, you can define labels by preceding an identifier (7 characters at most) by a colon (“:”), then using the test (**-t**) command to jump to that label. Test checks if substitutions have been made.
- Multi-line commands: sed has several commands that work with multiple lines, treating them as if they were a single line with embedded newlines. These commands make it hard to read sed scripts.

- Holding space commands: sed is capable of saving and exchanging the current line with a separate line, called the holding space. What is true for multi-line commands is even more true for commands that handle the holding space—they truly make your scripts unreadable.

As I said earlier, use sed for one-line commands and simple scripts and awk and Perl for advanced commands. For those who wish to earn a virtual beer: write a sed script that computes the factorial function.

> Hans de Vreught is a computer science researcher at Delft University of Technology. He has been using UNIX since 1982 (Linux since 0.99.13). He likes non-virtual Belgian beer and is a true globe trotter (already twice around the world). He can be reached at J.P.M.deVreught@cs.tudelft.nl.

Archive Index Issue Table of Contents

Advanced search

Letters to the Editor

Various

Issue #60, April 1999

Readers sound off.

LJ Enterprise Solutions

Thanks to Marjorie Richardson for mentioning KDE in the article “Linux and Enterprise: A Winning Combination” in the January supplement to *Linux Journal*. As initiator of the KDE project, I would like to add a few things.

KDE is far more than a desktop, as she seems to indicate. The main targets are not only users of the standard components (like the window manager or the startup panel), but also developers. Although it contains a window manager, file manager and a startup panel (what users usually call “Desktop”), this is only half the story. KDE is mainly a sophisticated application development framework. It makes it possible for programmers to create better applications for Linux in a much shorter time.

The main reason Linux has suffered from few and poor graphical applications was no free technology was available with which to build them. Every free software developer had to reinvent the wheel over and over, starting with a configuration system and a help browser and ending with a printer driver or at least a PostScript engine. KDE's success and the ever-increasing number of KDE-based applications demonstrate impressively what effect a desktop standard makes on productivity, and—even more important—the fun of programming. Note that “desktop standard” refers to the library level, object model and common helper applications; it has nothing to do with the way the windows are decorated or how users launch applications.

Thus, KDE is one answer to the “applications, applications and more applications” cry she mentions. I am talking about real applications here, including usable versions of a PowerPoint-like application (kpresenter), advanced vector drawing tools (killustrator), a document processor (KLyX), two

advanced spreadsheets (ksiag and kxcl), a state-of-the-art web browser (konquerer), news reader, mail client and many smaller utilities such as image viewers, multimedia tools, games, text editors, et al.

Please have a look at http://koffice.kde.org/ and http://www.kde.org/applications/ for more information.

—Matthias Ettrich ettrich@troll.no

Hope you liked the article in February's issue, “KDE: The Highway Ahead” by Kalle Dalheimer —Editor

LJ in Brunei

I cannot believe what happened to me two days ago when I went into a bookshop in the country Brunei, Darussalam, where to my knowledge I am the *only* Linux user. What did I see in the computer magazine section? A *Linux Journal*!

Thanks for giving me a great magazine even in a country without Linux.

—Stefanus Du Toit sdt@ultracool.net

Calendar

I liked the article “Take Command: Calendar Programs” by Michael Stutz (*LJ*, January 1998), so I installed **calendar** on my machine. With my system (Red Hat 4.1 and tcsh), I got “unterminated string” errors from entries like “John's birthday”. This is a consequence of the method by which GNU **cpp** manages quotes. I found a good workaround is to include this line:

```
(calendar >/dev/tty) >& /dev/null
```

in my .cshrc file instead of just **calendar**. It does not fix the errors, but it does throw them away.

—Tony Sumner solon@macaulay.demon.co.uk

Misstatement by *LJ* Editor

First of all, we at IGEL wish the *LJ* staff a happy, healthy and successful 1999!

In issue 57 on page 32 (“1998 Editor's Choice Awards”, Best New Hardware—Corel NetWinder), Ms. Richardson states, “Corel Computer is the first company to declare Linux ...” This statement is simply *wrong* and misleading. IGEL was

the first company to introduce a Flash Linux-based Network Terminal/ Computer or Thin Client back in 1994. *Linux Journal* even tested the unit at the time and printed a very positive product preview. This unit, then called Etherminal 3X/4X, is now replaced by the Etherminal W/J Thin Client series, which began selling much earlier than Corel's NetWinder. The functionality of Etherminal W and J goes beyond that of the NetWinder and has a much better price/performance ratio.

I would appreciate a printed correction of this statement, since our product marketing uses this time and functionality advantage publicly, and we do not like having to continuously argue against her statement.

On page 75 (“New Products”) is a short announcement about the release of Etherminal J, and we are very thankful for this mention. The following statement was cut from our press release:

> In addition, Etherminal J supports Citrix MetaFrame and Citrix WinFrame via the integrated ICA Client for Linux, and Tekcentric WinCentric via an integrated WinCentric Client for Linux. RDP support is planned to be released in the future as well. This makes Etherminal J the only universal “All in One” Thin Client Workstation available.

In the same article, the price of the product is now outdated. The new pricing is strictly based on quantity, ranging from $590 to $899 US. Reseller pricing is available.

Last, I'd like to offer an article on IGEL as a company, supporting and using LINUX as its only OS for any product offered now and in the future. We have some very interesting product plans which will literally take Linux and IGEL's Flash Linux technology to new highs in 1999.

Thank you for your interest and continuous support!

—Hans L. Knobloch President and CEO, IGEL LLC

My apologies for my misstatement with regards to the NetWinder. The Etherminal was indeed first. I would like to talk to you more about your company. —Marjorie Richardson

Re: Linux in Lebanon

These comments are about the article entitled “Linux in Lebanon” by Ibrahim F. Haddad which appeared in the January 1999 issue.

I was pleasantly surprised to see *LJ* cover Linux usage in our country and commend Mr. Haddad on a nicely written article. I would, however, like to point out that company usage of Linux has been somewhat downplayed. Our company is almost exclusively a Linux shop, and we have created very large web sites which are served on Linux machines.

For example, LebHost (http://www.lebhost.com.lb/) is a comprehensive search engine about anything related to Lebanon, and our main web site offers free web sites to Lebanese (http://www.greencedars.com.lb/). Both sites are Linux-based and very heavily visited. We also have dozens of hosted domains on Linux machines as well.

In our case, it was clearly not a matter of “follow the crowds”, even though Microsoft software is “freely available” in a pirated form here. We find Linux much more attractive to use because of performance, flexibility, superb support and source code availability.

—Edmond Abrahamian, PhD. edmond@greencedars.com.lb

Csound Article

I read the article in February's *LJ*, “Linux Csound” by David Phillips. I think you should add a link in the “Resources” section to the Quasimodo Project, http://www.op.net/~pbd/quasimodo/.

This project is a rewrite of Csound for UNIX, supporting multi-threading, with enhanced real-time performance, GUI, modular, etc. A first ALPHA (without GUI) is available now. When this project is finished, it will be absolutely the coolest Software Synthesizer System that has ever existed.

—Thomas huber@iamexwi.unibe.ch

SSC's Distribution Choice

I am a little confused over which distribution to get and install on my machine. All seem fine for most tasks. I remembered in a past issue of *LJ* that SSC uses the Debian distribution. Why was Debian your choice over other distributions? Was it because it is an all-volunteer distribution versus commercially-based distributions such as Red Hat, Caldera and others?

Additionally, if I choose a distribution other than Red Hat, I fear I may not be able to purchase software products because companies are specifically alliancing themselves to support only Red Hat instead of all major distributions (or, rather, certain components used in all distributions). This is reminiscent of Wintel systems versus Apple. More software was made to support Windows.

People, like vendors, will go where more is offered. Distributions that are not Red Hat compliant will be left in the dust—a situation that will lead to factions within the Linux community. Companies need to provide products without requiring Red Hat be installed.

—Jean Tellier jtellier@cts.com

Yes, our primary reason for choosing Debian was that it is a volunteer effort. Red Hat is part of the effort to develop standards for Linux that will keep applications from being distribution-specific. While many companies have ported their products to Red Hat first, I don't think any have said they will not support other Linux versions. Informix came out for Caldera first but now works for the others as well. Corel's NetWinder comes installed with Red Hat, but Debian also works well on it. I think you can pick whichever distribution you wish without worry —Editor

Non-root Shutdown Possible with sudo

I saw a question in *LJ* January 1999 (“Best of Technical Support”) for which I can provide a suggestion. The utility **sudo** can be configured to provide non-root shutdown capability. **sudo** (su do) basically allows non-root users to execute a restricted set of commands. It essentially performs an **su - root -c *command*** but can ensure more restrictive access control and does not require you to give out the root password.

On my desktop, I configure it so that any time I want to run a root command (I log in under a regular user account called stephen), I just type **sudo *command***, and it executes it for me without hassling me. Of course, you can configure it to require passwords, allow only a single command, etc.

More about sudo can be found at http://www.courtesan.com/sudo/.

—Stephen Thomas stephen@thomas.mitre.org

Name Misspelling

You accidentally spelled Jaroslav's name wrong in your article about Csound in the February issue of *Linux Journal*. The correct spelling is Jaroslav Kysela. (You were very close!) Thanks.

—Chris David cdavid@umich.edu

Linux for Macintosh

I have just finished reading Alan Cox's article “Kernel Korner: Linux for Macintosh 68K Port” (January 1999). An excellent article in an outstanding publication. I wait quite impatiently each month for *LJ* to appear in the mail. Please keep up the broad spectrum of articles—something for the experts and something for us beginners. Thanks.

—Ron Phinney rphinney@xroadstx.com

Archive Index Issue Table of Contents

Advanced search

More Letters to the Editor

These letters were not printed in the magazine and appear here unedited.

---

business plan

Just a quick question (worth printing to next issue?)

Is there any (good) GNU licensed programs to help the creation of businessplans?

—Jussi Kallioniemi [jukal@teraflops.com]

Invitation to Join Freedom of Choice Project

TO: Representatives of... OS user groups, consumer advocacy orgs., ISD associations, software vendors and distributors

FROM: James Capone, of the Linux Info Website
Diane Gartner, Co-ordinator of IACT

The Freedom of Choice Project is a co-operative effort between IACT and James Capone, an IACT member as well as devoted user of Linux, who created the entire project at his own Linux website. In the project's first week, over 5000 people had participated in the Freedom of Choice consumer poll. With help from IACT, James Capone now is expanding the poll to reach users of _all_ platforms.

As you know, all computer users certainly are affected by an ongoing problem in the computer market: Microsoft still maintains an exclusive distributorship with PC makers such as Compaq, Dell, Gateway etc.. Those companies pre-install or “bundle” MSFT software on the majority of new PCs we buy. Once the MSFT software is pre-installed, we may decide to delete it and then fight to get a refund, but that approach still won't get to the root of the problem.

The Freedom of Choice project is our grass-roots, long-term solution. By using the Internet as it was designed—to bring together small groups like ours into a larger, stronger and unique network—we're going to defend the fundamental right of consumers everywhere to choose any and all software that is installed on the new computers they buy.

We want to give users of all platforms the chance to _send a direct message_ to the PC makers, to demand that the companies fully respect every consumer's right to choose. But we need your help to do it!

Join us as *Project Associates* of the Freedom of Choice project. In the spirit of teamwork and co-operation, we are asking you to sign your name to this project, and make a strong commitment to helping the world's computer users regardless of platform, background, nationality or expertise.

The Next Step

Please r.s.v.p. to this letter and tell us if you can commit your organization's resources and name to the Freedom of Choice Project. If you cannot help in an active role as a Project Associate, then please at least consider posting links and announcements for us.

Your active participation would include a few, simple volunteer responsibilities:

1. Devote a small amount of your web space to a “Freedom of Choice” page on your group's website, to outline the goals/benefits of the project and to post frequent updates on the project's growth. Just be sure that your page gives equal emphasis to all OSs and to all computer users regardless of platform, background, nationality or expertise.

2. Directly contact your own members + associates, urging them to visit your website's Freedom of Choice page and then to respond to our Freedom of Choice Poll, so that your group's concerns about software choice will reach the PC makers with maximum impact.

3. Help us to publicize the project by using your media, political and IT contacts. Forward the project's official announcements to your contacts, and urge them to support the project either directly or indirectly.

4. Any other resources or ideas that you'd like to offer....

If you choose to join us as a Project Associates, then with your permission we will add your name(s) to this same Invitation when we send new copies to more organizations; your group likewise will be given a spotlight on our own Freedom of Choice web pages, for the public to see.

The Freedom of Choice Poll is at http://www.angelfire.com/biz2/Linux/company.html

regards,
Diane Gartner, dgwhiz@earthling.net
James Capone, linuxos@iname.com

Great Article
I really enjoyed reading Alessandro Rubini's article “Software Libre and Commercial Viability” in the February issue.

It is often hard to see where things are going in the Linux and Open Source world, and Mr. Rubini's article was very intelligent and enlightening with quite a few insights into what's up.

—Robert Lynch
rmlynch@best.com

Re: University of Toronto WearComp Linux Project (Feb 99)

I've used Linux since 0.99, and taken *LJ* since somewhere around issue 6 or 7. PLEASE PLEASE don't get political on me. The recent article by Dr. Mann filled the first 3.5 pages with his personal libertarian paranoid philosophy as it pertains to the use of intelligent devices in society.

I am happy for him to tell us about his project, how Linux allows him to be “COSHER”, and to describe his applications of Linux. However, I don't (and don't think most *LJ* subscribers) give a whit about Dr Mann's personal politics. To read the first three pages of the article, I suppose I should start taking apart my telephones, TV's Microwaves, and temperature-sensing shower heads because there may be a hidden videocam inside them.

Keep up the good work. I love your mag.

—Bill Menger, William.M.Menger@usa.conoco.com

Corrections

I've been subscribing to L.J. for a few months now, and I eagerly await each edition.

I think the magazine needs to dedicate more resources toward editing and reviewing the magazine for errors.

As an example, the latest edition's squid article states Netscape's proxy server does not support ICP, which is incorrect.

I think the quality of the magazine could be greatly improved if the editors worked more closely with the authors. As Linux gains in popularity,

I'm hoping the advertising dollars come in to enable you to continue to improve the overall quality (and width!) of each issue.

—A Demanding Fan, Jim Ford, jford@aetna.com

Re: BIS

Neil Parker wrote:

> Read Joseph Pranevich's article on Linux 2.2 in Kernel Korner (*LJ*: December 1998) with interest. He mentions support for a 'subset of the Linux kernel' on 80286 and below machines. On this

> tack I was wondering if there is anything geared towards making effective use of ageing 386 systems with 4Meg or so of memory. I have a lab full of these all running Windows 3.1 and would like to convert them to Linux if possible. Probably there are many thousands of similar machines worldwide.

No, not that I know of. That's a bit out of bounds for ELKS. (Linux for 8086-80286)

Your best option here is to compile a main Linux kernel, even a 2.2 kernel with the absolute minimum requirements for your hardware. There are also patches floating around for reducing memory requirements even further, but I can't give you a pointer right now. Having done that, you should strip down the a system to the *bare* essentials. No daemons, no loadable modules, nothing that isn't absolutely necessary. You'll also want to recompile your X server (Mono, SVGA, or VGA16?) with the minimum options and no unneeded drivers.

Of course, this will all have to be done on a bigger Linux system, recompilations on a 386 take *forever*

With all of that done, you'll get a working system. You may need to hack the X source a tad to not configure things like unneeded mouse drivers and etc. With any luck, that will fit, but barely. If you run all applications after that remotely, it might not be so bad.

Make sure you have swap space, Linux 2.2 actually runs faster with it because it swaps some unneeded bits out. And I've heard that Linux 2.2 would be faster than Linux 2.0 but you might want to check both. Alternatively, you could look into building it with a Linux 1.0 or 1.2 series kernel, if your hardware would allow it. And building a libc5 (as opposed to glibc) system might save you a tad more room. And finally, make *sure* that you have no static libs and compile everything shared.

I hope this helps.

Joe Pranevich, jpranevich@lycos.com

letter to editor

At the UW I work with all platforms but for ease of development I've used Linux since .99pl33

While working on a Journal article (for a medical publication) on computer security, I bought a copy of WinNT Magazine (Dec. 1998) and came across some comments about Linux that I think are unfounded, but I would like some expert and authoritative rebuttals.

/******

On page 35, Craig Barth writes

“According to intelectual property lawyers, the Linux licensing agreement binds any developers who produce software using components of the Linux OS (eg. libraries and runtimes) to release the source code for their additions. This will stop mainstream commerical development dead in its tracks”

On page 122, Mark Russinovich writes

“... whereas WinNT and all commercial UNIX implement kernel mode threads, Linux does not.”

“... the schedular cannot preempt the kernel ... Because Linux kernel is onot preemptable, it is not as responsive to high priority processes as other kernels are ...”

“... the Linux kernel is not reentrant, which means that only one processor in a multi mode system can execute kernel code at one time”

“ For the next couple of years, Linux is stuck with being only a valid choice for small uniprocessor servers ...”

****/

If you could respond with some sources, I would send such back to WinNT magazine.

thanks,
—Steve, sglanger@u.washington.edu

Intel Red-hat for the LAST time

Several months ago I wrote to you expressing my concerns over certain packages being made available just for users of the Red Hat distribution package. When this letter was published in the January issue I though I might get some feed back but I was amazed to see in the Feb. issue a letter by Fred Nance praising Red Hat.

I think you have all missed the point!

Yes, Red Hat and all the other distributions are doing very positive work in promoting LINUX, Without Red Hat several packages would not have been written and we would have definitely have a smaller user base and certainly not a significant force against Micro Soft.

How long will it be before one large company sides with Caldera, one with Suse and one with Red Hat and we all start to make incompatible packages? If we don't start pulling together soon we will destroy everything already built up and MS will be laughing all the way to the bank.

Could Red Hat give me a positive statement that all their packages including Oracle are compatible with other distributions and they don't use their own libraries and why was glibc2 picked up by Red Hat before it was formally released forcing incompatibility issues?

Thank you for your time.

—Bob Weeks

BTS correction: shutting down a Linux machine

I was just reading the “Best Of Tech Support” in the *Linux Journal* 1999, and was struck by the “Shutting Down” question, and its answer, since I have a habit of “unorthodox”(?) ways of shutting down.

Among other things, I don't particularly like the ctrl-alt-del method of shutdown, mentioned as “the only way... for any user to safely shut down a linux system is to be physically present at the keyboard and press ctrl-alt-del.” I've found that the odd DOS user will sometimes reboot a system because their window is not responding, so I sometimes disable it in inittab. Also, if this is a “headless” machine with the console on a serial line, there may not *be* a console to type ctrl-alt-del on.

Here is a reasonably simple, and relatively secure way of solving the problem (better checking of the arguments might be wiser). Compile the following, called shutme.c and set the permissions as below (with “user” replaced by the GID of the user, or of a login group allowed to use shutdown:

```
#include <unistd.h>

main(int argc, char **argv)
{
  int i;

  for (i=0; i<argc-1; ++i)
    {
      argv[i] = argv[i+1];
    }

  argv[argc-1] = 0; /* execv linkes a null-terminated list of args */

  execv("/sbin/shutdown",argv);
}

-r-sr-s---   1 root     user        4186 Jan 25 01:31 Monty_only/shutm
```

For paranoia, I put this in a directory which was chmod 700 and owned by the user in question, too.
Then, assuming that directory is in the user's path, they can run

```
% shutme -rf now
```

without being root, on most unix machines I've used, including Debian 2.1, and almost certainly Red Hat. A nice thing about this is that any flags to “shutdown” are available, including the ability to cancel a running shutdown.
Irix has some pickiness about that, and if Red Hat is similar, then there are some other options.

It's easy for any root process to initiate a shutdown by sending INIT the right signal; replacing the above with just

```
main()
{
  execv("init 6",0); /* or execv("telinit -t 10 6",0); */
}
```

or, if only shutdown is picky, execv(“reboot”) or execv(“halt”) should be just as good. The disadvantage of these is that they don't issue a wall to all the users, but that can be included in the program as well. Running sync just before is traditional, but init should take care of that without a hitch in modern times.
There are a few other options which *should* work, and do for some linux versions:

Make a group of people allowed to run shutdown, say “shutters”, and

```
# chgrp shutters /sbin/sutdown
# chmod 550 /sbin/shutdown
# chmod +s /sbin/shutdown
```

and then anyone in “shutters” should be able to run shutdown.
Lastly, if you make an account called shutdown whose UID is 0 (root) and whose shell is a shell script that runs “shutdown -rf now” (or whatever), you can give people that password to that account and they can

```
% ssh host -l shutdown
```

or

```
% rlogin ssh -l shutdown
```

or

```
% su - shutdown
```

and so forth.
I have used all of the above on various unix systems, in various states of security and/or partially-crashedness... I was thinking about it because I had to reboot a half-wedged SGI which I didn't have the root password for, but did have root on an NFS server whose disks it was mounting recently...

Anyway, not meaning to be too picky, but with a name like “best of tech support,” I think this answer fell below standards. I did learn both about the “cp --one-file-system” flag from your column this month, so I *do* appreciate it, by the way, and the info that all zip disks come as partition #4 was an interesting confirmation of a trend I've noticed.

anyway, hope this helps, Thomas...

—Mark Montague, monty@muggy.gg.caltech.edu

Letter to the Editor ...

On the same day that I received my *Linux Journal*, I got a PC Connection periodical. It's a standard PC mail-order catalog, but it seems to assume that people are running Windows. Are there similar catalogs that cater to people running Linux/Unix. Like have non-PnP modems, cards that include drivers for other OSes besides Windows, etc., etc.

—Charles Wheeler, snowwolf@sprynet.com

Article Suggestions

Just a quick suggestion for two articles on subjects that are sorely needed in the Linux community in my view.

Some of us users have some old DOS and Windows applications that we still use and there are no Linux alternatives. This makes us having to use a dual boot system.

Now I have had partial success with Wine in using a specialized communications program, but I have some DOS apps I would like to run under emulation that are indeed simple programs and should be amenable to run under Dosemu. I have not had success with Dosemu in 4 months of trying. I have not been able to find a source that simply explains how the program works and the theory behind how to get it running. The readme files, in my opinion, are very rudimentary.

We need some comprehensive articles on how to set up these applications. Now I know Dosemu has the reputation of just being a hack that so many people are using to run games but there are people like me who want to be able to do some useful work with Dos apps, not necessarily games.

I've had some success with Wine with a specialized communication program that I use to access my patient database at the hospital I work for. It is an old Windows 3.1 app that runs good with a little distortion of the fonts. It is still very usable and negates the need for me to have to boot to Windows every time I need it. One neat thing is that it uses the modem under Windows to make a 9600bps connection to the hospital server and under emulation, I do not see a speed hit whatsoever even though it is running through the extra layer of emulation.

I would hope you would see fit to find an appropriate author(s) who could tackle this task and I believe you would be doing a great service to the Linux community in helping to make Linux more mainstream in being able to run Dos/Windows apps in a Linux system.

I think the time is right to carry this out as I feel Linux is reaching “critical mass” and if a person can see that they do not have to give up on being able to run Dos/Windows programs by migrating to Linux, it might only give an extra boost in acquiring more Linux users.

You have my permission to edit this to your standards if you see fit to publish it in the *Linux Journal*.

Best regards,
—Kurt Savegnago, M.D., ksaves@prairienet.org

No more whinings, please.

I read, from time to time, about user friendly configuration tools for Linux. The last time I did read about is the Davis Brians Letter on *LJ* # 58 (feb. '99, pag. 6 “Linux Installation and the Open Source Process”). I'm sorry, but I totally disagree with the idea that such a tool is essential, nor useful. The Linux community, the real one, may need tools that help to spread the knowledge about Linux to neophytes, but don't need windozian tools at all. Those that need Bill's nightmare simply must stay with windoze. Linux is the evolution of UNIX (even if not the only one) and it is light-years far from MS, both in power and flexibility. People not sufficiently skilled to install Linux from himselves has at least three choices: 1) to buy a pre-installed system; 2) became a more skilled one or 3) stay with their best silly O.S. Everyone is free to get his own way but, please, don't blame others for your incompetence: Linux may be free (or Open Source or whatever you like to call it) but in the real life nobody can have hothing without some personal effort.

—Franco Favento, Franco_favento@Generali.com

Intel and RH

I have read Robert's letter on page 94, *LJ* Jan 1999.

I definitely agree with you that we don't need the Intel-Hat, or sth else which is another M$ WinDos alike OS.

In the long terms, it maybe comes to the end that there is only one distribution of Linux surviving in the market, but right now, I hope all the exisitng distributions work ahead, as this will bring the benefits to all Linux users.

In fact, all Linux distributions have the same problems: or says, the aims - they need easier way to install/maintain, and more the better GUI-based applications for business.

Best,
—Frederic, hongfeng@public.wh.hb.cn

re: Dear ljeditor. This little poem might amuse your readers....

I am a M$ separatist, hooked now on Linux. The reason I switched is described in the following little poem:

Flamed on the GPS newsgroup (? by Bill Gates), response in rhyme

Somebody said that I lied
That's bullshit, I quickly replied
An honest mistake
Can anyone make
I'll explain, and then you decide

My ThinkPad and Windows were wed
At the factory in the same bed
They shipped 95
And to keep it alive
An upgrade to 98, I ded

98 was a great deal much wurse
I could tell right away it was cursed (some kind of f.king virus)
It choked very hard
When I tried a sound card
The factory defaults I restured

The most outragous abuse
When Netscape I tried to use
Explorer stomped in
And GPF'd me like sin
My computer I had to rebuse

My times not worth much anymore
I'm just an old profes*sor*
With me its my health
Compared to his wealth
His geek thugs are what I deplore

When at the young age of 23
Keats got a bad case of TB
He then wrote some lines
That have help me define
What a loss Gates handed to me

(“When I have thoughts that I may cease to be,
Before my pen has gleaned my teeming brain”)

Better spent my own time would be
Writing for pos-ter-i-te

So, I've tried to put it in rhyme
Think about this in your spare time
And then you will see
What my time's worth to me
To take it away was a crime.

—M David Tilson, mdt1@columbia.edu

Is Linux getting too commercial?

I have a total of four PC's, two at home and two in my office. I have in both places one PC with an Uncle Bill Gates' system for talking with electrical engineers, and one PC with Linux system for talking with physicists. I have decided to keep these machines separate, since Uncle Bill has constructed a windowing system with an internal search and destroy program. This program is automatically run as soon as another system (e.g. Linux) is detected on the hard disk. Do not bother to tell me how to overcome such programming games, since I so much enjoyed viewing this program on a friends PC. In an unplanned (by the users) civil war program, Windows 95 took out NT.

However, I would like to express concern about Linux. After looking over the shoulder of a student depositing RedHat 5.0 on an office box, I easily enough installed it again on a box at home. Later I bought some RedHat “power tools”. I ignored the warning on the box that I would need RedHat 5.2. Then I had to go out and buy RedHat 5.2. My discomfort is not yet my anger at Uncle Bill for changing the file format in Microsoft Word, thus increasing the price of talking with electrical engineers. But really, ... 5.0 ... 5.2?

I am perfectly willing to pay a reasonable price for packaged software, without using up my students valuable time searching the overcrowded web sites for the free versions. But I do see some bad handwriting on the wall.

Further, when you merely update from 5.0 to 5.2, rather than doing a fresh install, things that used to work can stop working. My postscript printer file thought the printer moved from America to Europe. It changed the internal view of the paper size and forgot to tell me. The menu bars for increasing the console window size no longer show any words, but I can use the menus. I merely have to remember what the menus used to read when they actually had words on them.

I am afraid that when Intel buys into Linux, the commercial outlook will lessen the quality of the system. Can you get ruined by success?

—Allan Widom, widom@acausal.physics.neu.edu

Re: Small Linux Machines (CyberFlex_isn't_ a Linux machine)

Hi Jeff Alami,

Regarding your statement in 32bitsonline article “Small Linux Machines”

> “Which system has the bragging rights of being the smallest Linux computer around? My guess is Schlumberger's CyberFlex Open 16K smart card with the Linux kernel. The computer chip on card stores card holder information and even biometric information for secure authentication. Now there's a small Linux machine.”

I think you are mistaken here. According to the documentation on Schlumberger's CyberFlex web site, this is a smartcard that runs Java programs on a Java Virtual Machine under control of a small operating system called GPOS (General Purpose Operating System). The notion that the card runs Linux seems to have started with Marjorie Richardson's claim to that effect in the January 1999 issue of *Linux Journal* when granting the 1998 Editor's Choice Award to the Cyberflex. I don't know whether she has issued a correction to her statement since I haven't seen the February 1999 issue of *LJ*. I'll copy her on this message in case others haven't pointed this out to her.

So it looks like the smallest Linux machine is the one developed from off the shelf parts by Stanford University Wearables Lab. Now _there's_ a small machine that's actually running Linux.

Regards,
—Ronald L Fox, rfox@dls.queens.org

> See http://www.linuxnet.com/smartcard/index.html the MUSCLE web site for more about Linux and smart cards.

Reply to 8/98 Article!

In an 8/98 letter Mr. M. Leo Cooper states that one can Symbolically link the Netscape cookies file to the NULL device, thus preventing Heinous WebBots access to this info. (a Most Laudable Goal!)

Mr. Cooper's soultion is thus:

```
ln -s ~/.netscape/cookies /dev/null
```

Mr. Cooper has a Great Idea, on My System; however it Fails, for some Subtle reasons. (ln) will Fail IF: The Second File EXISTS ! In other words, as Printed, IF /dev/null Exists, then the Link will Fail ! Important: On Most Unix / Linux systems /dev/null, and /dev/zero are Required for Proper Operation, therefore, Likely to have been Created and Exist.
All is Not Lost though, here is how I was able to Accomplish the job:

1) Delete the ~/.netscape/cookies File, copy it to a Backup if you Want the Info, such as: cookies. Sav or cookies. Bak, Then delete cookies!

2) Use this form for the link command:

```
  ( ln -s /dev/null
 ~/.netscape/cookies )
```

This procedure worked Flawlessly for Me, /dev/null is Preserved, and a New @cookies Link is Created under ~/.netscape !
To Test my theory, I logged into a server that I knew was Particularly Nasty about setting cookies.

After the session I viewed cookies using the editor in Midnight Commander (a Favorite), and cookies displayed Absolutely Nothing, the Void that we Want that Uninvited WebBot to See !

This procedure is a little more involved than just piping something like:

```
 ls -l /home/cookies > /dev/null
```

But Well Worth the effort as “Big Brother” is snooping Relentlessly, and people have a Right to be Concerned for their Privacy!
Now, what do we do about [Pentium III (c) Intel Inc.] so called “Hardware Cookies” ? As an engineering student and systems programmer I, and some colleagues are discussing it!

Sincerely,
—Jim Boedicker, sysdev@swbell.net

spelling error(s).

On page 82 of your March 1999 issue, in the “Red Hat LINUX Secrets, Second Edition” review, you have 3 spelling errors, or mistyping. In the first column, around the 4th line down, it says “Linux kernel 2.2.35”, i believe that that is mistyped. I think it should be 2.0.35. Also, in the second column, 17th line down, it says “(2.2.35)” again, and 5 words later, “(2.2.32-34)”. I'm pretty sure Duane Hellums, the author, didnt mean to type these wrong versions. I just thought i would bring it to your attention.

—scott miga suprax@themes.org

article corrections

Message-ID:
<Pine.SO4.4.05.9902161119210.9359-100000@colossus.csl.mtu.edu>
I just read the review of “Red Hat LINUX Secrets, Second Edition” on page 82 of the March 1999 *Linux Journal*, and I'd like to point out a few problems with the article:

(i) Every time it mentions the kernel, it incorrectly refers to the versions as 2.2.x, when they should be 2.0.x.

(ii) It says, “Also helpful would be a loadable module for sound card support to avoid having to manually configure and rebuild the kernel...” I haven't purchased the book, but if it indeed comes with Red Hat 5.1, as the article claims, then the kernel is already preconfigured for sound card

use and installs all the modules that Red Hat supports. Red Hat also provides a nice tool, 'sndconfig', to configure your sound card and modify the /etc/conf.modules and /etc/isapnp.conf file (if needed).

(iii) It says, “...Linux is in dire need of an intuitive, commercial-quality, freeware, GUI-based word processor...” While technically not freeware, both Corel's WordPerfect and Star Division's StarOffice are available for free for individual, non-commercial use. KDE also comes with a text editor that provides approximately the same functionality as Wordpad.

—Jeff Bastian, jmbastia@mtu.edu

Freeware vs. mega$ware

Being in the position in my company to “affect the purchase decisions”, I have encountered some very disturbing trends in the way corporate world sees and uses IT equipment. As many of you out there I have been a victim of the “main-stream” systems which means for example that a $3,500,000 installation by Sun known at the company as “mega-server” every once in a while gives messages saying “No more users allowed to log in”. You know - the licensing thing. The database on which the livelyhood of the company depends is something of a joke. In fact that's a bunch of databases designed by different vendors that handle various aspects of production. They can neither be modifyed nor discarded due to the nature of contracts with their vendors and the sheer amount of money invested in them. So I am currently involved in a project of designing a super-database that would join them all in one working mechanism and would provide a usable interface to the whole system.

You would probably say what can be easier all the tools are avilable for any platform one would choose. Not so fast. The company went shopping for yet another vendor to build the database. The final choice was between (no-no, there was no mentioning Linux!) a company that would charge us $20,000 for the whole project and a company that would charge $500,000. (There's no typos in the zeroes, I am talking twenty thousand vs. half a million dollars.) Now try to guess which company won the gig. The half-a-million dollar one. Of course. And I voted for it too. Why? Because the bigger budget means more “petty-cash”, more restaurant invitations, more business trips etc. Besides some people in the company are dependent on the commissions... What kind of commissions do you get off a Linux-based job? $500, at most.

All of this makes me very depressed. There is no way in the world that Linux can make its way into corporations like this one. But something happened recently that gives hope. I saw a demo by Silicon Grafics of their new Intel-based workstations. They come in two flavors: NT and Linux. And SGI offers full industry-standard support for these machines. Which is great, because they are not cheap (forget about free).

What I am trying to say is, to make it into the corporate world we may need not just commercial, but extremely expensive systems. Like $3,500,000 servers.

Advertising

Regarding Chuck Jackson's letter in the March 1999 issue (in response to Mr. Havlik's letter), arguing that advertising can easily be skipped, will this may be true, skipping the ads always interrupts the flow of reading.

The same holds true for letters and articles which are “ ... continued on page ...”. I don't see any reason why e.g. the letters to the editor aren't on two consecutive pages in the magazine, but instead are separated by almost ninety pages (in the March 1999 issue). I hope *LJ* will try to avoid this in the future.

I know that most magazines can't do without advertising, however I would think that the main goal of *LJ* is to convey information about new trends to the linux community - not every ad in *LJ* matches this criterion.

Personally, I subscribed to *LJ* because of the informative articles, not because I like the advertising so much, but perhaps there's a way out for all of us:

I'd like to see a separate section packed with ads, e.g. in the second half of the magazine - just like the german c't magazine - which reduces the number of ads in the first half. This has advantages for both groups of readers, the ones who read the articles and the ones who are interested in the ads. Articles would be mainly in the first half and would contain less intrusive advertising (e.g. ads would only precede or follow the articles, not interrupt them).

—Lars Michael, lmichael@empros.com

Issue 59 editorial comments

Let me briefly state that I have been a loyal subscriber since around issue 5 and thoroughly enjoy *LJ*.

I'm a bit disturbed by your comments to Reilly Burke who wrote the “Red Hat Phenomenon” letter.

“ . . . However, Red Hat does seem to be the most popular distribution available, so they must be doing something right.”

This sounds exactly like the answer I get when I try to convince users of Evil Empire software that there are alternatives which are more robust and feature rich. Reading between the lines, I get the impression that any distribution which gains market share is a good thing and if one dominates, it must be the most technically sound. By this logic, we should all dump Linux and run one of the flavors of Windows. We should also dump Emacs and LaTex and only use MS Office tools. Clearly they own the largest market share and are therefore are doing the most thing right.

A year or two ago I probably would have agreed but we have seen an astounding increase in Linux coverage in general media. Not just

technical publications devoted to Unix but a broad range of publications from Byte Magazine and Computer Shopper (both slanted towards MS) to *Dr. Dobb's Journal* to mainstream non-technical publications. This is mainly due to the announcements by major vendors to support Linux (Sybase, Informix, Corel, etc), and of course Netscape's decision to open the source to their browser.

I personally use Red Hat and have been happy with it. I will say that I would not rule out switching to a different distribution if it seemed to have advantages.

I am also a bit confused as to what Reilly means by “conventional Unix methods”. I have been using Unix for a number of years and have always thought that there were two main Unix flavors; SysV and BSD. Vendor specific Unix implementations are typically based upon one of these base OSs.

I hope this does not sound too harsh, it is not meant to be. I think you and all the staff at *LJ* have been doing a great job.

—John Basso, jbb@msd.ray.com

popular does not equal good

Reilly Burke's letter (March 1999) criticized Red Hat and you responded that Red Hat is the most popular distribution so they must be doing something right.

Well, Microsoft must be doing one hell of a lot of things right in that case, but that does not imply that what they produce is of high quality.

Actually, I did not understand Reilly Burke's criticism of Red Hat as 'difficult to install'. Red Hat 5.2 installed for me with zero hassles. Nor his remark about Red Hat having a 'strange implementation' (unclear what that means). As a UNIX user since version 6 in 1975 and a Linux user since 1995, first using Slackware, now using Red Hat, I have always found all UNIXes to be arcane but very sound. Usually, when someone is criticizing a given UNIX or given text editor, or a given whatever, it is because they are used to something else and are not making room for the learning curve.

That being said, we all know that we have a task ahead of us in developing easier install shells both for the Linux OS and for Linux applications, as well as continued work to agree upon a File Hierarchy Standard and Control Panel for GUI-based System Administration.

regards,
—Dennis G. Allard, allard@oceanpark.com

editors reply

I the March 1999 issue your reply to Reilly Burke is quite assinine:

“Sorry, you will have to ask Red Hat about their policies. I am not in their confidence. However, Red Hat does seem to be the most popular distribution available, so they must be doing something right. —Editor”

Sorry, why do you think people read your journal but to get information. For you to put someone off the way you did Reilly Burke is about the most pitiful reply I have ever seen. It makes you seem too lazy to ask Red Hat about it (what your subscribers expect you to do) and like an ass for answering in a flippant manner.

WindowsXXX seems to be the most popular distribution, actually, they must be doing something right, you betcha, they market very well, however the software they release seems to me and large numbers of others to be nearly unusable because of stability and frustration caused by difficulty of use.

The man has a legitimate question about Red Hat, so that leads me to conclude that you are (as most journals are) only interested in the advertising revenue.

—Gene Imes, gene@ozob.net

Archive Index Issue Table of Contents

Advanced search

A Look to the Future

Phil Hughes

Issue #60, April 1999

My goal was to have *Linux Journal* address as many of the needs of the community as possible.

In 1993, before I began publishing *Linux Journal*, I knew that in order to succeed the magazine would need the support of the Linux community. At that time, most of the community existed on the Internet—not surprising, since Linux development was primarily on the Internet and most developers had met only over the Internet.

To test the waters, I posted a questionnaire to the comp.os.linux newsgroup on Usenet. I had two goals: to find out how much interest existed for a print publication and what information was needed. Interest was high; I received very positive feedback. My goal was to have *Linux Journal* address as many of the needs of the community as possible.

The first need was to help newcomers join the Linux community. Many were not familiar with Usenet or didn't have access to it. These newcomers needed an accessible source of information in a convenient format—a print magazine addressed that market.

Introducing businesses to Linux was another area that had to be addressed. While our very existence was a help—many people have told us they managed to get Linux integrated into their business by showing their boss a copy of *LJ* to prove Linux was real—we wanted to be more active. We started our “Linux Means Business” column to show *LJ* readers where Linux was being used as a business solution.

The Community Evolves

Today, the Linux community is quite different from what it was five or six years ago. A strong development community still exists, but the business and

commercial user community is now a significant portion of the total Linux community and growing rapidly. Linux is no longer a development project—it is a real solution for a significant number of enterprises.

The Jay Jacobs retail chain is installing point of sale systems based on Linux in their 180 stores. The U.S. Postal Service has installed 5000 OCR systems based on Linux to scan mail. Also, Linux is being used to run elevators in Japan, trains in Germany and interactive TV in Denmark—even to mediate reality in Canada.

Is there still a community here? Yes. Its members aren't sales clerks in Jay Jacobs stores or postal workers bagging mail, who most likely don't even know they are using Linux. They are the people porting software to Linux, IT managers selecting Linux and those installing Linux. These people need to work together and learn from the work of others.

To help build this community, *Linux Journal* has done two things. First, I volunteered to coordinate a mailing list for Independent Software Vendors (ISVs) as part of the Linux standardization project. This list is designed to give ISVs an opportunity to have a united voice in the standardization effort. Second, we added an IT Solutions supplement to *Linux Journal*, designed to help IT professionals see Linux as a possible solution to their needs. The first supplement appeared with the January, 1999 issue of *LJ*; the next will appear with the June issue.

Two Communities?

We are, in fact, building two communities. If this was happening with proprietary software such as Microsoft products, I would see this as a problem. In the Linux realm, however, I feel all is well. Let me explain.

The Linux development community benefits from growth in the commercial use of Linux. This growth also means more hardware and software vendors will want to support Linux, resulting in more potential employment for Linux professionals.

On the other end, commercial users of Linux benefit from growth in the development community. Unlike proprietary software, more developers means better software without increased cost.

These two communities complement each other. Their levels of interests or understanding may not all be the same, but one benefits the other; hence, cooperation is needed between the two.

Linux Journal has plans to aid in that cooperation. Just as we provided a forum for developers and newcomers in the early stages, we hope to provide a forum for the development and commercial communities to better understand each other. We are addressing this goal by doing the following:

- IT Solutions supplements, help for business professionals wanting to know what Linux can do for them
- Linux Resources (http://www.linuxresources.com/), a starting place for finding all you want to know about Linux
- *Linux Journal Interactive* (http://interactive.linuxjournal.com/), containing serious articles on Linux from both a technical and a business point of view
- *Linux Gazette* (http://www.linuxgazette.com/), an open forum for discussion of technical issues, as well as articles and Linux tips

SSC, the owner of *Linux Journal*, publishes books and reference cards to help people use Linux and Linux-related programs. Examples include books on Samba and the GIMP and a whole series of reference cards to help the more technical user deal with utility programs included in Linux distributions. In 1999, expect to see new titles to help Linux move to the desktop. More information can be found at http://www.ssc.com/.

The bottom line is we have been a part of the Linux community since 1993 when SSC first began selling Linux distributions. The Linux community and *Linux Journal* have grown up together, and we plan to remain the primary print resource for Linux as its evolution continues.

Phil Hughes, Publisher

Archive Index Issue Table of Contents

Advanced search

New Products

Ellen Dahl

Issue #60, April 1999

Cyclades-PR4000, PerlDirect, ICS and more.

Cyclades-PR4000

Cyclades Corporation announced a new member of its Power Router family, the Cyclades-PR4000, a powerful, compact and cost-effective remote access server. It allows Internet Service Providers and Enterprise Network Managers to terminate both analog and digital calls and provide network access to remote offices, telecommuters and home users. The Cyclades-PR4000 is scheduled to ship in April. Contact Cyclades for pricing.

Contact: Cyclades Corporation, 41934 Christy St., Fremont, CA 94538, Phone: 510-770-9727, Fax: 510-770-0355, URL: http://www.cyclades.com/.

PerlDirect

PerlDirect provides reliability, stability, support and accountability for Perl through the following features: validated, quality-assured releases of Perl and its popular extensions; advice and support; Y2K test suite; and a Perl Alert weekly bulletin. PerlDirect offers an opportunity to provide direct input to a leading organization involved in open-source development. Basic annual subscription rates start at $12,000 US.

Contact: ActiveState Tool Corp., PO Box 2870 Main Station, Vancouver, BC V6B 3X4, Canada, E-mail: sales@activestate.com, URL: http://www.activestate.com/.

ICS

BASCOM announced the availability of its Internet Communications Server (ICS), an educational software/hardware solution developed for the OpenLinux OS from Caldera Systems Inc. for K-12 schools. BASCOM's use of OpenLinux provides the education vertical market with its first Linux-specific application. The ICS includes an access management engine, firewall security, proxy cache and software subscription service. ICS is available through BASCOM as a fully integrated, drop-in solution, as well as from BASCOM's OEM partner, Caldera Systems. Pricing for solutions in this line starts at $1,995 US.

Contact: BASCOM Global Internet Services, Inc., Phone: 888-922-2726, E-mail: info@bascom.com.

Linux Network Server Package

CTiTEK announced a new Linux network server installation. Server package includes labor (estimated 50 hours) and license costs (full version of Red Hat 5.2); hardware cost not included. Price is $3349.97 US for a 25-user Linux network server package.

Contact: CTiTEK, 14377 Woodlake Dr., Suite 311, Chesterfield, MO 63017, Phone: 800-860-9913, 314-878-9855, Fax: 314-878-9893, E-mail: sales@ctitek.com, URL: http://www.ctitek.com/.

CSM Proxy Plus for Linux Version 4.1

CSM Proxy is a full-featured proxy and caching server, especially written to handle the various demands of network administrators. It can solve Internet connection needs in one simple package. Key features and benefits include SMTP command filtering, IMAP 4 proxy, HTTP cookie filtering, connection limits, automatic Netscape proxy configuration, remote administration over the Web and much more. CSM Proxy runs with any web server and does not require a web server on the same machine. Price is $695 US; version 4.1 update is $100 US. (Users who ordered after April 1, 1998 can upgrade to CSM Proxy 4.1 for free.)

Contact: CSM-USA, Inc., 360 South Fort Lane, Bldg. 1, Suite B, Layton, UT 84041, Phone: 801-547-0914, Fax: 801-546-0716, E-mail: sales@csm-usa.com, URL: http://www.csm-usa.com/.

Empress RDBMS v8.10

Empress announced Empress RDBMS's support for both 32-bit and 64-bit applications on Hewlett-Packard's HP-UX 11. Running under HP-UX 11, the

Empress RDBMS offers application developers an environment which delivers high performance, reliability and optimized features for Internet computing. Empress Hypermedia, an Internet application development toolkit, is used for the creation of the actual web applications. Empress Suite for Linux includes Empress RDBMS (bundled with a C programming interface and Empress Report Writer) with host-level accessibility for enhanced performance tuning. Visit the web site for pricing or a free 30-day evaluation.

Contact: Empress Software, 6401 Golden Triangle Drive, Greenbelt, MD 20770, Phone: 301-220-1919, E-mail: info@empress.com, URL: http://www.empress.com/.

M-Cluster

Alta Technology Corporation announced the availability of M-Cluster, a compact and fully modular Linux-based clustered computing system for use in high-performance and complex data processing/analysis applications. The M-Cluster uses rack-mountable single-board computer and peripheral modules. M-Clusters are fully integrated with the Linux operating system, which implements Parallel Virtual Machine (PVM) and Message Passing Interface (MPI) technology. The Alpha, from Compaq, is currently available in M-Cluster systems with clock speeds of up to 500MHz. Pentium II processor versions will be available by the second quarter of 1999. M-Cluster systems are available with prices starting well under $20,000 US.

Contact: Alta Technology Corporation, 9500 S. 500 West #212, Sandy, UT 84070-6655, Phone: 801-562-1010, Fax: 801-254-4329, E-mail: sales@altatech.com, URL: http://www.altatech.com/.

LynxArray and LynxNSS

Artecon announced support for the Linux operating system across its LynxArray and LynxNSS product lines. LynxArray is Artecon's highly scalable, high-performance RAID product line for corporate data centers, scientific and

technical applications and environments, such as telecommunications infrastructures or on-the-move operations. LynxNSS is Artecon's network-attached storage (NAS) product line that supports network file system protocols for the UNIX/Linux and Windows operating systems for file sharing and storage over the network. Additionally, LynxNSS uses TCP/IP, the universally accepted Internet protocol, for network communication as does Linux. Contact Artecon for pricing.

Contact: Artecon, 6305 El Camino Real, Carlsbad, CA 92009, Phone: 800-872-2783, 760-931-5500, Fax: 760-931-5527, E-mail: sales@artecon.com, URL: http://www.artecon.com/.

GO-Global, GO-Between, GO-Joe

GraphOn's products are designed around open-system and client-server concepts for the multi-vendor environment. All provide high-performance access to UNIX/X and Linux applications anywhere on an organization's Intranet, the Internet or over dial-up connections. GO-Global 1.5 is a thin-client PC X server that delivers high-performance access to UNIX applications from any Windows desktop. It supports Red Hat and Caldera Linux. Evaluation copies of GO-Global are available from GraphOn's web site. Pricing begins at $295 US per seat. GO-Between is a thin-client PC X server that provides access from Microsoft's Windows Terminal Server and other multi-user NT solutions to UNIX and X Window applications, and supports Red Hat Linux. GO-Joe, a thin-client Java X server, works on any Java-enabled device or desktop to provide plug-and-play access to UNIX/X and Linux applications, and supports Red Hat Linux. Contact GraphOn's sales department for pricing of GO-Joe and GO-Between.

Contact: GraphOn Corporation, 150 Harrison Avenue, Campbell, CA 95008, Phone: 408-370-4080, Fax: 408-370-5047, E-mail: sales@graphon.com, URL: http://www.graphon.com/.

Linux Main Memory Database Benchmark

Polyhedra has developed a benchmark demonstrating main memory database performance on Linux. Features include an active query mechanism that ensures automatic data change update from a Polyhedra server to its client; rapid application development and easier change/modification through the use of object-oriented software techniques; event-driven interfaces to both high-speed data devices and to other commercial databases and systems; systemwide SQL access. Visit the web site for information on pricing and support.

Contact: Polyhedra, Inc., 1611 - 116th Ave. NE, Bellevue, WA 98004, Phone: 425-646-4907, Fax: 425-646-3020, E-mail: sales@polyhedra.com, URL: http://www.polyhedra.com/.

Magnate Internet Store

ParaSoft announced the launch of their fully scalable, e-commerce solution on Red Hat Linux. Internet Store is a fully automated, commercially affordable e-commerce system which can automatically update a retailer's actual inventory when an on-line sale is made. It provides a virtual shopping cart and quick and easy pricing, as well as secure credit card transactions. Whenever a retailer adds new items to his inventory or makes any pricing changes, Internet Store publishes the new information automatically on the store's web site. When a customer makes a purchase online, Internet Store instantly notifies the retailer of the sale. Call for pricing.

Contact: ParaSoft Corporation, 2031 S. Myrtle Ave., Monrovia, CA 91016, Phone: 888-305-0041, Fax: 626-305-3036, E-mail: info@parasoft.com, URL: http://www.parasoft.com/.

LinuxCare, Inc.

LinuxCare, Inc. provides technical support, consulting, product certification and education to the business, government and education industries. It offers a unique web-based service with an easy-to-use search interface to Linux support resources. The company was co-founded by David Sifry, a contributor to the Linux kernel and VP of the Bay Area Linux Users Group. LinuxCare is committed to the concept of open-source software.

Contact: LinuxCare, Inc., 6034 Fulton Street, San Francisco, CA 94121, Phone: 888-546-4878, Fax: 415-831-9763, E-mail: info@linuxcare.com, URL: http://www.linuxcare.com/.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #60, April 1999

Our experts answer your technical questions.

Accessing /dev/fd0

I successfully installed Red Hat v5.1 with X as the GUI. I have O'Reilly's book *Learning Linux*, which is a good reference. The only problem I have is that I cannot access /dev/fd0. When I type this command in an xterm, a message states that access to the device is forbidden. What am I supposed to do? I have looked up information on the Linux USENET listing, but I cannot find anything specific. —P. Kincaid, pkincaid@osf1.gmu.edu

You either need to be root to access /dev/fd0 or belong to a group (e.g., floppy) to which users with access permissions to /dev/fd0 belong. On my system, it looks like this:

```
brw-rw-r--  1 root  floppy  2,  0 mai  5  1998 /dev/fd0
```

—Marc Merlin, marc@merlins.org

The simplest way is to run the following command as root:

```
chmod 666 /dev/fd0
```

Bear in mind that this has security implications, especially on a multiuser system. —Scott Maxwell, s-max@pacbell.net

Hidden config File in X

When I want to start the X Window System, the system shows error messages—the first few lines look like this:

```
Could not find config file!
 -Tried:
      /root/XF86Config
```

```
        /etc/XF86Config
        /usr/X11R6/lib/X11/XF86Config.slackware
        /usr/X11R6/lib/X11/XF86Config
    Fatal Server error:
    No config file found! ...
```

—Fazli Yusof, fazliona_14@hotmail.com

You have not set up X on your system; it is installed, but not configured. A good way to get started is to run xf86config. This program will step you through several questions and then set up a simple XF86Config file that should get you up and running quickly. Once you get X running you can make any other changes you might need. —Chad Robinson, chadr@brt.com

Upgrading an Old Kernel

I want to upgrade my kernel. The one I am currently using is a few years old. Can I upgrade all the way to 2.0 or 2.1? One of the things I need in a new kernel is support for the CMD 640 chip. Right now, Linux can see only one of my hard drives. I am also looking for something that will play audio CDs from my NEC 260 2X CD-ROM. A few more specifics: I downloaded kernel 2.0.33 (I think). It has the support for the CMD chip. I can get all the way through **make config**, but then I get errors when trying to compile the kernel. It tells me I need gcc 2.6.3 or above. I got 2.7.2 from an FTP site and installed it along with the current libraries. However, now when I type **gcc**, it says it cannot execute the binary file. What's up? —Jim Coonradt, coonmanx@yahoo.com

A lot has changed since your system was brought up. Linux went through three binary formats (a.out, elf/libc5 and elf/libc6 aka glibc). There is a lot to fix and upgrade on your system to get a recent kernel to compile and run properly. If you are interested in doing so, you can look at /usr/src/linux/Documentation/Changes/ on a recent kernel, and learn about the three binary formats, but be prepared to do a lot of reading. A much easier alternative is to back up your user data, and install a brand new Linux distribution. If you are interested in running 2.1 and the upcoming 2.2 kernels, you should consider installing Red Hat 5.2, Debian 2.0 or higher, or SuSE 5.3. —Marc Merlin, marc@merlins.org

Wrong Date

The date on my Linux box is not correct. My time is GMT+5. I set the universal time so it starts by subtracting five hours from it. My current date shows this:

```
# date
Sun Jan  3 05:17:39 GMT+5 1999
# date -u
Sun Jan  3 10:17:43 UTC 1999
```

—Bilal Iqbal, storm@compucraze.net

To set the timezone, you should make the following symbolic link:

```
ln -s /etc/localtime /usr/share/zoneinfo/US/Pacific
```

To set the time, you can do it relative to UTC or your local time; it depends on what time you stored in the BIOS' clock. Read the man pages for **hwclock** (or **clock** if the first doesn't exist) for more information. —Marc Merlin, marc@merlins.org

When Not to be root

When I use TELNET to log in to my system, I cannot log in as root (I get the login incorrect message). However, I can log in as myself and use **su** to become root using the root password. What's up? —Scott Hall, shall@ccae.org

The file called /etc/securetty defines the terminals a root user may log in to. Make sure ttyp0, ttyp1 and so forth are defined in that file. Each TELNET session uses one of these terminals, so define several. If you don't and normal users are logged in on p0 and p1, you will not be able to log in as root on p2. —Chad Robinson, chadr@brt.com

TELNET is set up to deny root access as a deliberate security measure. This keeps a cracker from compromising the root account directly—he has to first compromise a user account, thereby making it twice as hard to become root on your Linux box.

Incidentally, the potential cracker's job is made even harder by not indicating that no password will work for root when logging in remotely. The cracker can't tell whether his login attempt is failing because he has the wrong password, or because remote root logins are entirely disabled. —Scott Maxwell, s-max@pacbell.net

Shells and a.out vs. ELF

I think they are somewhat similar so I'll group my questions:

How can I tell if I have a.out binaries on my system? Any mention of these suggest them to be an old format. Is there anything still around that I'm likely to be using that requires a.out support in the kernel?

Also, how can I tell which shell(s) I need on my system? It seems several get installed, but I'd be happy to stick to just one. Do I need to scan for scripts that may use something other than the default shell? If so, could someone suggest an appropriate scan script to demonstrate “The Power” to a newbie, please?

Perhaps the savings are modest for each of these, but so are the resources for my foray into the Linux world! —Barry Johnson, barryjj@ibm.net

To search for a.out binaries, type:

```
find / -mode +400 -exec file {}\; | grep MAGIC
```

*The **find** command looks for files with the executable bit set. **file** prints what they are and **grep** extracts a.out files (identified as either ZMAGIC or NMAGIC or similar strings. System scripts use only /bin/sh (no csh, zsh or other shell). On the other hand, some scripts may be written in other languages, and their interpreter is sometimes called shell. These dependencies, however, are managed by your packaging system. If you are curious, repeat the **find** command line looking for —Alessandro Rubini, alessandro.rubini@prosa.it*

On the first question: typing **file *progname*** will tell you whether ***progname*** is ELF or a.out.

Listing 1

As for the second, Listing 1 is a short Perl script that prints the shell and file name of all executable shell scripts on your system. Run it as root. The shells are printed first so that you can pipe the script's output through **sort** to sort the results by shell; this helps you quickly skip past the shells you decide to keep. It will take some time to run and probably will produce a great deal of output, so you should redirect the output to a file. —Scott Maxwell, s-max@pacbell.net

Paging or Performance Questions

This is my first Linux system and I'm having trouble tracking down some information. Maybe you can help me out or point me in the right direction.

1. I have Red Hat 5.2, Netscape 4.5 and cable internet access. My FTP throughput is excellent, but general browsing on the Internet is almost unbearable. My 9.6 modem could be faster (well, almost). It's kind of disappointing, compared to my NT system. Anyway, my first question: I hear regular disk activity, every five seconds or so. It's like some kind of paging is going on. What do you think is going on in the background? If I leave the PC alone for 15-20 seconds, it quiets down.

2. My second question is: while a web page is loading, I can start up **top** in an xterm window, move the window around and actually get faster downloads. I can't explain it. I can be sitting with a partially loaded page for 30 seconds, then bring up top and it's like Linux woke up and started updating the screen. —Daryle Dianis, ddianis@home.com

1. I take it this disk activity isn't linked to a **cron** job, but indeed to paging. A good way to make sure is to install and run **procmeter**; it can show you all kinds of activity meters, including disk activity and paging.

2. You don't give any details about your hardware, but it sounds like some kind of hardware and/or driver problem. Make sure you are running the latest version of X (3.3.3.1), and you don't have any interrupt conflicts. For example, you may have an interrupt configured for your video card (you don't need one) that conflicts with the interrupt of your network card. You can also try changing the interrupt of your network card. —Marc Merlin, marc@merlins.org

This may be a long shot, but do you have power management turned on? If your hard drive is spinning up and down, it might be producing this type of problem. Anything that represents more frequent activity (like top) that kept your drive from spinning down (or your CPU from slowing, etc.) would seem to speed up your computer.

An interesting thing to try would be to go into your BIOS at boot time and turn off all of your power management features. If that turns out to be helpful, you can turn it back on one feature at a time until you isolate the issue. —Chad Robinson, chadr@brt.com

Archive Index Issue Table of Contents

Advanced search

DECnet Network Protocol

Steve Whitehouse

Patrick Caulfield

Issue #60, April 1999

This article contains information on how to use and configure available DECnet software as well as information on how the kernel code works.

DECnet was designed by Digital as a way to interconnect their range of products. In its Phase IV implementation, released in 1983, it can support 63 areas of 1023 nodes each. The specifications for DECnet Phase IV are freely available (see Resources), which has allowed others to provide DECnet connectivity in products such as Sun's Sunlink DNI and Linux.

In networking terms, DECnet is an old standard. Its limited address space is far less than that of TCP/IP and it does not have the advanced features of more modern networking standards. DECnet is still widely used in legacy systems, and the intention of the Linux DECnet project was to allow integration of those systems with Linux-based solutions.

The Linux DECnet project intends to support only Phase IV DECnet, since previous versions in current use are very limited in number these days. LAT, another network protocol designed by Digital, will not be supported because it is covered by patent protection and its specifications are not freely available. In this article, we will use the term “DECnet” to mean the DECnet Phase IV family of protocols.

The DECnet Family of Protocols

DECnet can be carried over a variety of different data link layers. In the beginning, the Linux kernel DECnet layer will support only the Ethernet link layer; support will be added later for other link layers such as PPP, DDCMP and X.25. The PPP link layer is described in RFC1762 and the others are described in the DECnet documents (see Resources).

Like many network protocols, DECnet can be viewed as consisting of a number of software layers. More details are included in the section called “A Tour of the Kernel Sources”.

At the top of the stack is the application layer, which includes all the programs used on a day-to-day basis. These programs use the system libraries and system calls to create connections to other nodes. The kernel socket layer interface and the system libraries encompass what the DECnet standard refers to as the session control layer. It performs largely the same function as the library and system calls for TCP/IP. Further down is the network services protocol (NSP), fairly close to TCP in function. Below these is Routing, which does more than routing; it is a kind of IP and ARP rolled into one. At the bottom of the heap is the actual device over which the data is transmitted.

Each machine, called a node in a DECnet network, is identified by an address consisting of a 6-bit area number and a 10-bit node number. These two numbers are written separated by a dot, so 1.2 is a computer in area 1 with a node number of 2. Unlike TCP/IP, the address refers to the computer, not the interface through which communication is to take place.

Kernel Configuration

Two different sets of patches are available to add DECnet support to the kernel. The currently available code is based upon a patch written by Eduardo Serrat

for the 2.0.xx version kernels to function as an endnode. In parallel with this, one of the authors, Steve Whitehouse, was also writing a DECnet layer with an emphasis on creating a router implementation.

The result is you can now get the original patch for 2.0.xx kernels written by Eduardo Serrat (a version of the same code ported to the 2.1.xx kernel series) and also another patch which has modifications by Steve to make use of the newer support functions in the 2.1.xx kernel series. This last patch will be distributed as an add-on for the 2.2.xx kernel series and later integrated into the next development series.

Most of what we say here applies equally to all versions of the kernel patches. We will point out the differences as we go.

First, you need to retrieve the correct patch for your kernel. In general, the best way to do this is to get a copy of the most up-to-date kernel in whichever kernel series you intend to use. You can then download and apply the latest patch to the kernel source as described below. I also strongly suggest that you study the release notes for the kernel version you have, since the configuration procedure may change.

To apply the Linux DECnet patch, unpack the kernel source into /usr/src/linux as normal. Then obtain the correct patch for this kernel version and uncompress it in the directory above the top-level Linux source directory, /usr/src/ in this case. Then type:

```
$ patch -p0 < patch-file
```

patch-file is the name of the patch you want to apply. Next you need to compile a kernel in the usual way, being sure to say **Y** or **N** to the question about including DECnet support. Depending on which version of the patch you have, some of the options outlined below may be available.

The two main types of DECnet nodes are endnodes and routers; the latter is subdivided into Level 1 and Level 2 varieties. At the time of writing, only endnode support is available.

When DECnet router support is available, you will need to enable the option at compile time. Also, at module load or boot time, you will have to turn it on—a kernel with router support compiled in will be able to function as both an endnode and a router.

The DECnet *raw sockets* option allows the reading and writing of DECnet packets at a lower level than users normally need. It is very useful for debugging and monitoring activity, and might be required by future user-level

routing daemons. The main advantage of using this rather than a **PF_PACKET** socket is allowing a program to take advantage of the kernels filtering of invalid DECnet packets.

In order to use the DECnet kernel layer, you must also tell the kernel which DECnet address to use. This is the only point at which the instructions are dependent upon the kernel version being used. For 2.0.xx version kernels, you need the **startnet** program, which would normally be run in the boot scripts for your system. For the 2.1.xx version kernels and upward, the DECnet address can be set either on the command line with an option **decnet=1,2** or when the module is loaded. (That is a comma, not a dot, between the **1** and **2** in the previous sentence.)

Setting Up Ethernet Cards

Those of you familiar with TCP/IP will recall that the ARP protocol is used to allow a machine to discover the Ethernet address of other nodes attached to the network. No equivalent of this protocol exists in DECnet; nodes must have their Ethernet address set according to their DECnet node address.

In order to work out which Ethernet address to use, you take the four byte "hiord" prefix specified by the DECnet protocol and concatenate it with two bytes *xx* and *yy*, derived from the DECnet node address of the node you are configuring.

```
AA:00:04:00:
```

xx and *yy* are the least and most significant bytes of the 16-bit DECnet address, respectively. The bytes are ordered this way because the original systems upon which DECnet was implemented had little endian CPUs. Thus, a node with address **1.1** will have an Ethernet address of **AA:00:04:00:01:04** and a node with address **1.2** will have an Ethernet address of **AA:00:04:00:02:04**.

This address needs to be set in your Ethernet card before you start the card. On Red Hat systems, this is easy. You simply add the line

```
MACADDR=AA:00:04:00:02:04
```

to the file /etc/sysconfig/network-scripts/ifcfg-eth0 or whichever file corresponds to the Ethernet card you wish to use. If you are not on a Red Hat system, you will probably have to look through the startup scripts to find the **ifconfig** command for the relevant interface and add the options **hw ether AA:00:04:00:02:04** at a suitable place. If you are using Slackware, then /etc/rc.d/rc.inet1 is the correct file to modify.

If this seems too complicated, a utility called **dn2ethaddr** can be used to print out the Ethernet address of a node given the DECnet address on the command line. It can also be used within scripts; an example is given in the man page.

The File Utilities

The front end for the DECnet layer that most users will see is the file utilities, a collection of programs using the kernel socket layer to implement file transfer and other useful applications. Eduardo Serrat's original kernel patch came with a few example applications, which have been taken over by Patrick Caulfield and enhanced during the last few months.

Most of the supplied applications for DECnet use the DAP (Data Access Protocol) that performs a similar function to the FTP protocol in TCP/IP. DAP is one of many high-level protocols implemented on top of DECnet; cterm is another, which provides terminal access in a similar manner to TELNET on TCP/IP.

Files and File Names

The applications use the OpenVMS transparent DECnet file name format to refer to files on remote machines. This syntax should be familiar to OpenVMS users, although it may look a little odd to Linux users.

```
   nodename"
```

For example:

```
   tramp"patrick mypass"::[docs.html]art.html
```

The more eagle-eyed will notice that typing this file name into the bash shell causes it all sorts of problems because the shell has special meanings for quotation marks and square brackets. To get around this, we have to enclose the entire file specification in single quotes:

```
   dncopy 'tramp"patrick mypass"::[docs.html]art.html'\
   art.html
```

This command copies the file from the OpenVMS system to our Linux machine. If you're wary about having passwords visible on the command line, read the sidebar on DECnet proxies. Although not every DECnet file name you type in will contain special shell characters, it is a good idea to get into the habit of using the single quotes so that you don't get unexpected effects if you forget them when they are needed.

The syntax of file names on an OpenVMS machine is also a bit different from that in Linux. Directories are enclosed in square brackets and delimited with

dots. File names can have 39 characters on either side of the dot and both are case-insensitive. OpenVMS displays them in upper case but they can be referred to in lower case. The Linux file utilities will always convert the file names to lower case for you, since that is more convenient for Linux users.

OpenVMS treats a file as a collection of records rather than a stream of bytes. It likes to know how to delimit the records, whether they are fixed or variable length, and how to display them on the screen (carriage control).

The file utilities available in version 0.10 are:

- **dncopy**: copies files between OpenVMS and Linux systems.
- **dntype**: displays the contents of an OpenVMS file on standard output.
- **dndir**: displays a directory listing.
- **dndel**: deletes OpenVMS files.
- **dntask**: execute commands on an OpenVMS system.

dncopy

dncopy is the most complex of all: it uses what may seem to be a bewildering list of options. A file on Linux is simply a collection of bytes, whereas OpenVMS has a very rich file system. Files can have different organizations, record formats and attributes (see “OpenVMS File Types and Attributes”).

dncopy has to cope with the task of making sensible conversions between the “a file is a file is a file” attitude of Linux and the more sophisticated OpenVMS system. When copying files from OpenVMS to Linux, OpenVMS makes all the information about the file available as part of the network protocol, so this operation rarely requires a user to understand the nature of the remote file.

When copying a file to OpenVMS, the situation is more complex. **dncopy** has to tell OpenVMS what type of file it wants to create, what the record format will be and any other optional attributes that may be required. We have tried to make the default as useful as possible, so that if you copy a Linux file to OpenVMS, you get a useful file. OpenVMS has a file type that is analogous to Linux files in the SEQUENTIAL STREAMLF file. This is a sequential file you can seek with records delimited by line-feed characters: when you use dncopy to send a file to OpenVMS, this is what you will normally get. In fact, dncopy goes further than that and actually looks for records in your file when it sends it in order to make it meaningful to OpenVMS.

STREAMLF files are fine, but often you want to send block-structured data files or OpenVMS savesets that have been backed up or downloaded from the Internet, or perhaps you want your text files to be in the more normal

OpenVMS text file format. This is what all the complicated options in dncopy are designed to help you with.

A few examples may help illustrate. Normal OpenVMS text files have variable length records with implied carriage control. To send a file from Linux like this, we would type:

```
dncopy -rvar -acr myfile.txt \
   'tramp"patrick mypassword"::'
```

The option **-rvar** instructs dncopy to tell OpenVMS that the resulting file is to have variable-length records. **-acr** indicates that the records have implied (carriage-return) carriage control. Also notice the resulting file name has been left off. **dncopy** will use the base name of the source file (myfile.txt) in this case.

Alternatively, if you were sending a file to be used in a FORTRAN program, OpenVMS has a FORTRAN carriage control attribute, where the first byte of each record says whether to start a new line, new page, etc.

```
dncopy -rvar -aftn fortfile.txt \
   'tramp"patrick mypassword"::'
```

If you wanted to send an OpenVMS saveset (a bit like a Linux **tar** file), you would send a file with fixed-length records. The normal mode of dncopy is to send records, since records are what OpenVMS expects. Binary files have no real record structure so we must tell dncopy to send blocks of bytes and the size of those blocks. A common size for saveset files is 8192 blocks, so we could send a saveset file from Linux to OpenVMS with the command:

```
dncopy -mblock -b8192 saveset.bck\
   'tramp"patrick mypassword"::'
```

dncopy takes wild-card characters for both Linux and OpenVMS file names. (OpenVMS wild cards must be used for OpenVMS files: % for a single character and * for multiple characters.) As a result, you can copy whole directories at a time. It can also redirect by using standard input and standard output as destination files, with the hyphen as a file name. In this way, you can embed OpenVMS files in Linux shell scripts and pipelines.

One “feature” of dncopy you may never need but which grew out of its object-oriented design is that it will also copy Linux to Linux and OpenVMS to OpenVMS. Note that if you do an OpenVMS to OpenVMS copy, all the data will pass through your Linux box on its way.

dntype

dntype is really just a symbolic link to dncopy that forces it to send the file to standard output; it is really there just to provide consistency and save typing.

dndir

dndir is a directory command (quite like **ls** in Linux). It displays the OpenVMS directory in a format similar to the ls command. It takes a few switches to customise the format, though **-l** is probably the most used, as it displays most of the useful information.

Two fields that look different from ls are the file size and protection information. The file size is shown in 512 byte blocks and the file protection information is shown in OpenVMS format rather than Linux format. I chose to leave the protection display this way, because OpenVMS has more file protection bits than Linux and it is often helpful to be able to see all the information.

dndel

dndel deletes OpenVMS files. Like dncopy and dndir, it can take an OpenVMS wild card file name to delete multiple files. With the **-i** option, you will be prompted whether you really want a file to be deleted.

dntask

dntask is the only one of these programs that does not use the DAP protocol; instead, it communicates with an arbitrary DECnet object. One little-used feature of DECnet on OpenVMS is that by using the syntax **TASK=filename**, the command **filename.COM** will be run as a command procedure (the OpenVMS equivalent of a shell script) and the output can be redirected back to the calling task. Three example tasks are provided with the distribution. One simply issues a **SHOW SYSTEM** command which sends its output to the Linux machine (using the command **dntask tramp::show_system**). The output from this is analogous to the Linux **ps** command. Another sends the **-i** (interactive) flag to dntask to allow the user to interact with a shell on the OpenVMS machine. However, the following example is the main reason dntask exists.

Eduardo Serrat, who wrote the kernel layer for DECnet, made sure it was compatible with X11R6. This means that if you have DECnet support compiled into your X server (see http://linux.dreamtime.org/decnet/Xservers.html for pre-built X servers with DECnet support), you can start X Window System applications on an OpenVMS machine and have them display on a Linux machine. This is a cheap and efficient way to provide X terminal support for OpenVMS systems. The dntask program can issue a command to start any X

program to display on the Linux machine, provided you write a suitable remote command procedure. The example below shows a DECterm being started (something I personally use quite a lot), but it could also be used for more sophisticated things, such as starting a complete CDE session when a user logs in to Linux and starts X.

```
dntask 'tramp::decterm'
```

Other Utilities

Eduardo also provides the following useful DECnet utilities:

- **sethost** provides terminal access to OpenVMS machines, similar to TELNET.
- **ctermd** is a daemon that provides the opposite service, allowing OpenVMS users to **SET HOST** (or TELNET if you prefer) to a Linux machine.
- **dnmirror** and **dnping** are test utilities for checking that the software is installed correctly and verifying connections to particular OpenVMS nodes.

All the above utilities and the X servers depend on libdnet which is available from our web site (see Resources).

A Tour of the Kernel Sources

For the kernel hackers, here is a quick spin around the relevant source files. This section applies only to the newer kernel patches (i.e., for the 2.1.xx series and up), as those are the ones we expect you'll find most interesting and useful.

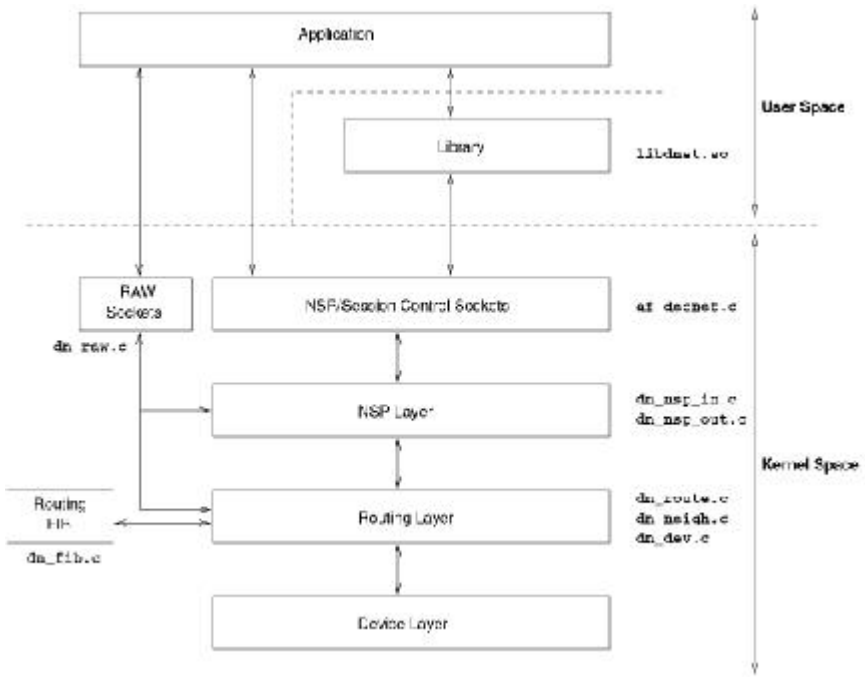


Figure 1. DECnet Software Layer Diagram

A diagram showing the overall layout of the Linux DECnet layer is shown in Figure 1. Where we describe the DECnet protocol, we go into just enough detail to give an idea of the main features of each part of the kernel code. If you want to know more about the way the kernel code works, you will need to read a copy of the DECnet specifications (see Resources) and then look at the source directly.

The main source file is af_decnet.c. This file contains the socket layer interface and parts of the DECnet NSP and session control layer code. Since the DECnet layering model does not map exactly onto the socket code, session control is provided partly by the kernel and partly by a user space library called libdnet.so.

dn_raw.c contains the code which implements the raw socket layer. It was one of the first things written, since it is very useful when debugging to see what is going on “under the hood”. It is also a good example of how to write the simplest socket layer interface possible. The file is compiled only when the raw sockets option is configured.

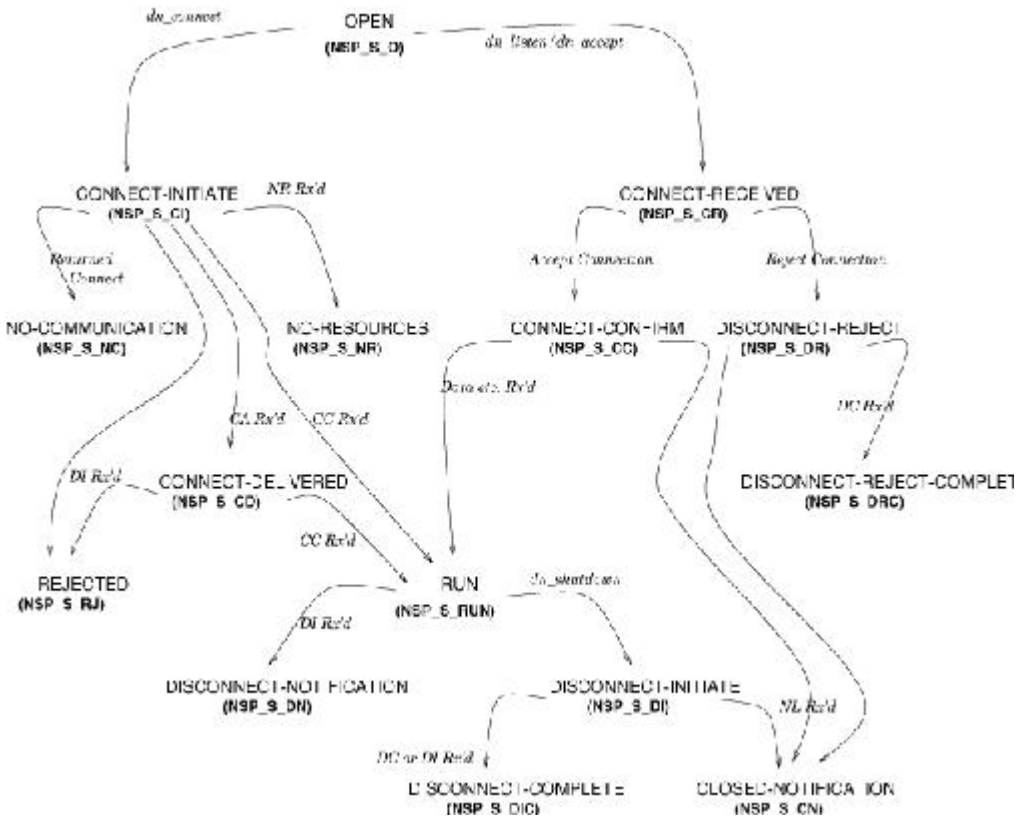


Figure 2. State Diagram of the NSP Layer

The rest of the NSP layer is divided into two parts: one for dealing with outbound packets, dn_nsp_out.c, and one for incoming packets, dn_nsp_in.c. The state table for the NSP layer is shown in Figure 2. We won't say much about the diagram here, but it should be a useful aid when used in conjunction with the kernel code.

The Routing layer is rather problematical. It has been divided into several files, due to the fact that the Routing layer actually does much more than just routing. dn_route.c is the main file which deals with incoming and outgoing packets, and dn_dev.c provides support for device-specific functions.

dn_neigh.c has a split personality. When a node is running as an endnode, it provides the On-Ethernet Cache described in the DECnet specifications; for routers, it provides the adjacency database. Since they are so similar, the decision was made to merge the two functions in order to keep the code small.

The actual routing functions (compiled only when the node is configured as a router) are in dn_fib.c. The code in this file is very experimental at this time, as decisions are still being made regarding how much of the routing should be done in user space and how much in kernel space.

Main Kernel Code Paths

One of the more obscure and important parts of the code is the main path for outgoing data packets. The DECnet layer uses the protocol-independent destination cache written by Alexey Kuznetsov, and neighbor table code written by Alexey Kuznetsov and Pedro Roque. These were originally designed to do some common processing required by the IPv4 and IPv6 network protocols, with the intention that other protocols would begin to use them at a later date.

What exactly do these two bits of code do? We will start by describing the neighbor table. The idea behind this is simply to keep a list of known nodes which are directly connected, along with certain information used by the protocol in question to communicate with them. In the case of TCP/IP, this means the ARP subsystem; for DECnet, it is used to hold one of two things. For endnodes, it holds the list of known nodes on the directly connected networks with which communication can be established, known as the On-Ethernet Cache. For routing nodes, it holds what the specifications describe as the adjacency database. In both cases, the function is the same but the actual method of operation is slightly different.

In the endnode case, hello messages are sent by routers every ten seconds to all endnodes that are directly connected. They are used by the endnode to create entries in the On-Ethernet Cache. Should a hello message not be received for a certain length of time, normally one minute, the entry is removed from the list. A default router is a directly connected node to which packets should be sent if the endnode is not connected directly to the destination. The default router is determined by information in the received hello messages.

For routing nodes, hello messages are received from both endnodes and other routers and are used to update the adjacency table. In this case, the entries are

removed if no hello messages are received for a length of time—a specified multiple of a time length noted in the hello messages. Currently, the neighbor table does not support different timeouts for each different neighbor. This problem is being worked on and may be solved by the time you read this.

One other piece of information held in the neighbor table is the format of header to be used by the routing layer in transmitting NSP data. There are two formats, one for use over broadcast links (long format) including Ethernet, and one for use over point-to-point links (short format). This is done by setting a pointer to a function to the correct routine **dn_long_outout** or **dn_short_output** when the table entry is created.

The destination cache is based on principles similar to the neighbor table. However, the object is to hold information required for each destination. When a packet is to be sent to a certain destination, it is looked up in the destination cache to see if it exists. If so, then that entry is used; if not, a routing algorithm must be called to discover the correct destination.

The routing algorithm also depends upon the routing or non-routing type of the node. The algorithm for routers has not been properly implemented yet, but will reside in the file dn_fib.c when that time comes. For endnodes, the algorithm is simply to send directly to any node in the On-Ethernet Cache, to send to the default router if it is not in the cache, or to send directly if there is no default router.

Again, a function pointer is available in each destination entry for a routine that will add destination-specific information to outgoing packets, then call the output routine of the neighbor.

That about wraps up the main features of the kernel code. There is, of course, a lot more to it than what is mentioned here, but we hope our overview will be useful if you're planning to add features or help with debugging. If you have specific questions, we'd be happy to try to answer them; however, please read the documentation first and also remember that we may not always be able to send an answer right away.

The Future

Hopefully, we have given you a good overview of the Linux/DECnet connectivity available at the time of writing. However, we are still hard at work on new features and programs (see below), some of which may be ready by the time this is printed.

dapfs is a file system layer for Linux which will let you mount an OpenVMS file system onto your Linux machine.

fal is a file listener for Linux, which will allow users on OpenVMS machines to access files on network Linux machines without having to log in.

Router support is also being worked upon. This is expected to take the form of a small amount of kernel code and a user-level daemon. It will allow you to connect multiple DECnet networks to your Linux machine.

DECnet Proxies

OpenVMS File Types and Attributes

Resources

Acknowledgements

Steve Whitehouse is a research student at the University of Cambridge, England. His research topic is “Error Resilient Image Compression”, and he is sponsored by Racal Radio Ltd. In his spare time, he contributes code and bug fixes to the Linux kernel network code and DECnet. Please feel free to e-mail him about Linux DECnet or his research topic at SteveW@ACM.org.

Patrick Caulfield is a software developer for the Santa Cruz Operation in Leeds, England. As most of his previous jobs involved at least 100 VAXs, he misses them a little and so got involved with the DECnet project. He lives in Leeds with Helen and six mad cats (who also have their own home pages) and can be reached by e-mail at patrick@pandh.demon.co.uk.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The Xxl Spreadsheet Project

Vincent Granet

Issue #60, April 1999

This paper is a general presentation of the Xxl project and of its latest version, describing the choices that drove the design of Xxl and its main charactertistics.

During the time frame of my academic work, I must periodically deliver statements of student grades to the administrative offices. The number of grades I manage is not immense, about two thousand per year, but it is enough to warrant automating the computations with the help of a spreadsheet program. My daily computer environment is Linux and until 1996, I was regretting the lack of a public domain graphical spreadsheet program for the X Window System, one which was user friendly and simple to use.

Of course, spreadsheets have been available in the UNIX world for many years, but in my opinion, none of them met the criteria of freeness, simplicity and user-friendliness I desired. I was forced to use a commercial spreadsheet on another computer with a different operating system in another office. This situation was not convenient, so I decided to build my own spreadsheet. At the same time, there was a call for proposals of computer projects in the Computer Science Master's program. This was a good opportunity to launch the project and I submitted this idea. It was chosen by a team of four students, thus, **Xxl** history began.

A Short History of Spreadsheets

The origin of spreadsheets is in the world of accounting. This name refers to paper sheets used for computing cost differences between offer and demand, and more generally, between two prices. These computations were made by hand and were tedious and error-prone. The first theoretical work on computer versions occurred in the early 1960s. The main principles of using matrices, cells and budget simulation, as highlighted by these theoretical works, were implemented in FORTRAN programs for simulating enterprise budgets (see Resources).

However, it was not until 1978 that two MIT students, D. Bricklin and B. Frankston, designed a computer spreadsheet, VisiCalc, that was truly usable on a personal computer. VisiCalc allowed many small enterprises to use costly computer techniques previously affordable only to larger companies. This program contributed significantly to the success of personal computers, especially the Apple II, for which the first version was written. VisiCalc was a tremendous success, but unfortunately for its authors, a very short one. At the beginning of the 1980s, VisiCalc encountered difficulties in trying to deal with the rise of IBM PCs. It was replaced by a new spreadsheet based on VisiCalc, Lotus 1-2-3 by M. Kapor. New features of Lotus 1-2-3, such as a simpler denotation for cells, the concepts of cell row or macro and the addition of graphic handling quickly made it a market success and a de facto standard.

In the late 80s, after Macintosh opened the door, most operating systems began to provide graphical user interfaces. The spreadsheets Quattro Pro and Excel appeared at that time and made sheet handling simpler, thanks to the use of tear-off menus, buttons and dialog boxes. Today, Excel is one of the most widely used commercial spreadsheets.

For many years, the UNIX world did not seem to be interested in spreadsheets. The first free and open spreadsheets available were **sc** and **ss** (an sc variant). They more or less offer all the basic functionality expected from a spreadsheet, but have no graphical interface. In the early 90s, Xspread (based on sc) and Oleo from the GNU project were the first spreadsheets with an X interface. Both provide new features (Xspread can generate graphics), but their graphical interface is still awkward and lacks user friendliness (none of them offers tear-off menus or provides for the X-style cut and paste).

Today, thanks to the qualities of its kernel and applications, the success of Linux is such that this system is beginning to overshadow some great builders of commercial operating systems. Linux is beginning to enter the industrial world—this is one of the reasons for the present burgeoning development of office suites (StarOffice, Koffice, Applixware, Gnome project's Gnumeric) which aim to equaling their commercial rivals.

Xxl has been designed for ease of use with a user friendly interface. It does not try to compete with spreadsheets such as Excel or Lotus 1-2-3, but will handle small and medium-size sheets.

Design

Most often, due to lack of time or competence, the programming projects of the Master's degree in Computer Science of the University of Nice—Sophia Antipolis are either not completed, or the final result cannot be used in practice. Expecting a usable result should not be a priority, since the goal of the

project is mainly pedagogic. For this particular case, I wanted to get a usable preliminary version of the spreadsheet, even though I anticipated some parts of the software product would need revision, as was the case. Thus, the goal of the project was clearly defined and the spreadsheet was limited to the basic functions of any classic spreadsheet, with a flexible and user friendly interface. The project was divided into two different parts, as little connected as possible. Two students dealt with the spreadsheet kernel and two others worked on the graphical interface.

Since the spreadsheet functionality was limited from the beginning, it had to be written to allow for subsequent extensions; therefore, choosing the programming language was of utmost importance. I wanted a language that would facilitate writing a spreadsheet prototype and enable an incremental development method. It also needed to be easy to alternate quickly between test cycles and corrections. Moreover, it should integrate an easy-to-use graphical library to free the programmer from all cumbersome aspects of X programming.

I chose STk, developed by E. Gallesio, which is an implementation of the programming language Scheme together with the Tk graphical toolkit (see Resources). In fact, it is similar to the Tcl/Tk system, with the Tcl interpreter replaced by a Scheme interpreter. More specifically, STk provides all the power and ease of use of the Tk toolkit from within the Scheme world. Moreover, Lisp (Scheme is a Lisp dialect) has already demonstrated its qualities for software extension (e.g., GNU Emacs). Finally, STk offers an object layer, STklos, which provides for a posteriori reutilization and extensions.

Tk (and thus STk) does not provide in its distribution any specialized widget for representing the computing sheet of a spreadsheet program and its predefined components do not make such a widget easy to build. The students could not devote more than half a day per week to the project. Asking them to program such a widget was out of the question—the project would have been an immediate failure. Thus, it was necessary to reuse some already-built components in order to make the programming task as small as possible.

One of the strengths of Linux is that it offers its users an open world. Thousands of programmers around the world are designing and developing programs, often of superior quality, which they offer for free to the international community. One of these programmers, J. Hobbs, is the present maintainer of a Tk widget called tkTable. Not only does this widget specialize in the representation of computing sheets, but it also had a property that made it the ideal interface between my two teams of programming students, i.e., the spreadsheet kernel and its graphical interface. The tkTable widget provides for associating a data structure (representing the cell values) with the graphic

computing sheet. After any change in content of the computing sheet, the widget automatically updates the data structure. In addition, and even more importantly, after any change in the data structure, the widget updates the computing sheet. Thanks to this property, the two student teams could work fully independent of each other.

The second component we re-used was the LaTeX environment, available on any UNIX platform. Xxl uses LaTeX for printing computing sheets and previewing them on the screen.

In June 1996 at the project's end, the students delivered the first usable version and we decided to distribute it freely. To give access to the source text of a software product is to accept the judgment of those who will scrutinize it. I used part of my vacation time to revise the code before the first distribution, which occurred in September.

The next year, to my disappointment, no student team chose to pursue the development of Xxl. Its present state was satisfactory for my own needs and I had no time to devote to its further development. Then I received mail from a new web site aimed at promoting scientific applications for Linux (SAL). They offered to give me a page for Xxl. This site gave the spreadsheet a larger audience, which encouraged me to improve it. Once more, I spent part of my summer vacation modifying the Xxl code, which I signed with a mock name. This time I rewrote almost all the code, with considerable simplifications, thanks to the use of the Flex and Bison parser generators. I also corrected some errors and offered new functionality. In September 1997, a new version was delivered.

At the beginning of June 1998, E. Gallesio delivered a new version of STk, with a small error which prevented the spreadsheet from working correctly. Since I was not using this version, I did not have a problem, but I got many messages asking me to correct it from users who thought the error came from Xxl. I had not been aware that Xxl had so many users. I devoted part of July to the distribution of a working version.

While it is clear that in an open environment like Linux, re-using already built software components is a good thing and should be encouraged, this does come with problems. In particular, programmers who use external components which they do not fully master (even with the source code at their disposal) are very dependent on the evolution of these components. Software tools like Flex, Bison or LaTeX, which have been stable for many years, did not cause any problem. This was not the case for the tkTable widget or for STk, which is still under development and constantly changing.

With each new version of tkTable, it is necessary to adapt it to STk. This represents about a hundred changes in the source code of the widget. Obviously, Xxl must follow the evolution of the tkTable widget, since the improvements are a benefit to it.

The evolution of STk could also be a source of problems. Fortunately, the author of this language is a friend whose office is not far from mine. This made it possible to solve many problems. While Xxl certainly benefitted from the improvements to STk, STk also benefitted from Xxl. In fact, the sheer size of the spreadsheet makes it a good test program, exposing some errors in the interpreter.

Characteristics of Xxl

Xxl has most of the characteristics of a classic graphical spreadsheet program. It handles computing sheets containing information stored in cells. Each sheet is edged with scrollbars. The number of sheets is not limited except by the size of the physical memory.

Computing sheets are structured in rows and columns. Rows are denoted by numbers and columns by letters. A cell is the intersection between a row and a column and is denoted, as in Lotus 1-2-3, by a column number and a row number (e.g., A1, AB23, ZZA2345). A cell range is denoted with the first and last cell, separated by a colon (e.g. A1:A3, A1:C1, A1:B8). Note that the last example denotes a rectangle. Users can interactively and without limits increase the number of rows and columns of a sheet. Every sheet is headed with a typing area for entering the value of the current cell. Figure 1 shows an example of an Xxl spreadsheet.

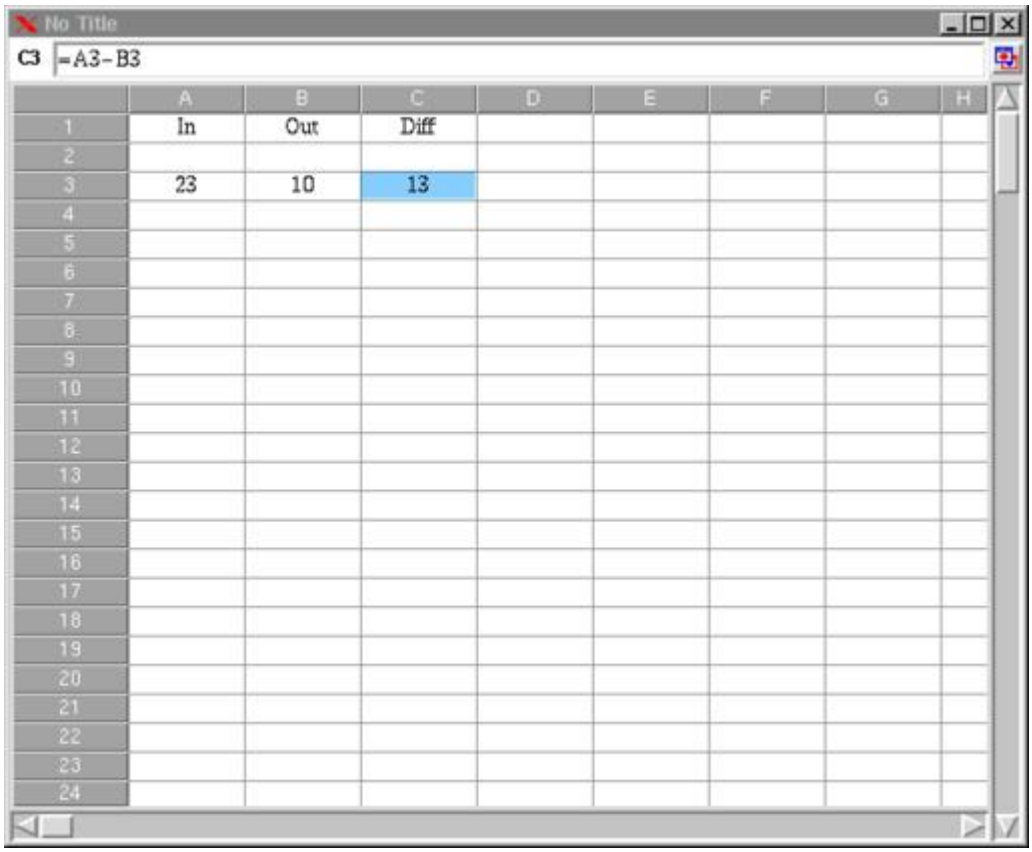


Figure 1. An Xxl Spreadsheet

The information stored in cells is made up of data (character strings, numeric values or booleans) or formulas which enable the computing of data. Integer arithmetic is of infinite precision. Formulas are mathematical expressions preceded by an equal sign. They involve arithmetic and classical relation operators, as well as a set of predefined functions. The functions deal with arithmetic (sum, prod, max, etc.), statistics (avg), logic (if, not, when), time (date, time) or character string handling (concat, len). Notice that all visible cell values are automatically computed anew.

Formulas can also contain internal or external references to cells in two modes: absolute or relative. A dollar sign in front of a row or column name denotes an absolute reference (e.g., $B2, D$13, A1). Absolute references do not change in move operations (addition or suppression of rows or columns, cut-and-paste operations). Without a dollar sign, the reference is a relative one (e.g., B2, D13, A1). It can be changed by a move operation. The cell C3 of Figure 1 contains the formula **=A3-B3**, which represents the difference between the values of the A3 and B3 cells.

The preceding references are internal ones, since they refer to cells in the same computing sheet. By contrast, external references refer to cells in other computing sheets and must be preceded by the name of the specified computing sheet. This name is that of the UNIX file containing the computing sheet.

Xxl offers several modes for displaying the cells. First, the size can be modified by enlarging or shrinking rows or columns. Second, Xxl offers several conventional modes for displaying the cell contents: several fonts in various sizes, bolding and slanting; several justification modes (left, right, centered), texts on several lines; several number representations (fixed, scientific, financial, percentage) or currencies (franc, dollar, euro).

One of the aims of Xxl is to provide a simple usage model by means of a graphical interface. All features of the spreadsheet can be accessed through a control panel, consisting of a bar with buttons, menus and a message area, all controllable by the mouse (see Figure 2). The control panel acts on the current sheet, which can be selected by the “sheets” menu or by a button in the upper left part of each sheet.

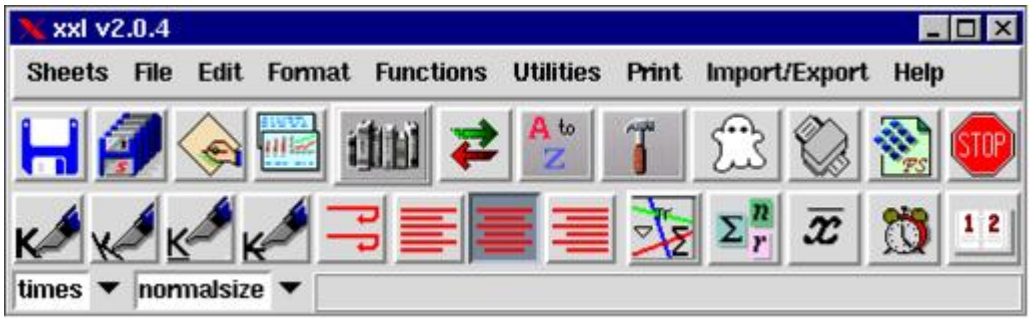


Figure 2. The control panel

The main functions to which the control panel provides access are the following:

- Selecting the current sheet: when several computing sheets are open at the same time, all commands of the control panel apply to the current sheet.
- Opening, closing and saving computing sheets: the storage format is the STk code that describes the computing sheet. Thus, loading a sheet is simply interpreting the program that describes it.
- Printing and previewing the computing sheets: these functions use LaTeX.
- Creating series: this function enables creation of a sequence of integer or textual values, with any step size.
- Cutting and pasting rows or columns
- Sorting rows or columns: this can be done by increasing or decreasing key values.
- Writing computing sheets in several formats: LaTeX, text, csv, HTML.

The mouse is also used for two important functions: selecting references and cut-and-paste operations. Entering cell references in a formula can be done automatically by selecting the needed cells with the mouse. This is especially

useful for references to rows or to external references, where keyboard input is error-prone. Copying and pasting cells within a computing sheet or between two different sheets is done according to two modes: whether one wants to copy (or move) values with or without formulas. Xxl also allows cutting and pasting from a computing sheet to another X application and vice versa.

Finally, Xxl offers on-line help with dialog boxes and balloon contextual help on the buttons of the control panel. On-line help, however, is somewhat scarce, but it can be considered sufficient if the spreadsheet usage is intuitive enough.

Perspectives

We have just considered the main characteristics of Xxl. One could ask which features are missing to make it a first-class spreadsheet. Xxl was designed to be simple and it will remain so. I do not think, for example, that it is necessary to be able to handle dozens of fonts or page layouts.

New features to be implemented in the near future are the following:

- Exporting to other formats for representing computing sheets (SYLK, WKS, ...) and importing files in these formats. It is very important, in my opinion, to ensure compatibility with other spreadsheets.
- Extending the set of functions for the formulas. Presently, the spreadsheet offers only thirty predefined functions. Search functions (hlookup, vlookup) are missing, as well as financial computations.
- Generating graphics. Histograms and pie charts are important tools for aiding quick and global understanding of a computing sheet.
- Internationalization. A Russian user noted that he was unable to sort character strings written in his mother tongue. In the last version of STk, E. Gallesio integrated some comparison mechanisms for character strings that use localization. Thus, internationalizing the spreadsheet will be done very soon.
- Integrating a mechanism to allow users to add their own extensions.

Long-term development will deal with making the production of computing sheets more reliable. Writing a spreadsheet is an error-prone activity. A very interesting study by R. Panko (see Resources) shows that a significant error rate exists in the production of computing sheets of various sizes, even when done by experienced developers. He also demonstrates the frequent lack of validation methods for building computing sheets and the scarce use of a posteriori checking tools. It is also surprising to learn that many companies use simple spreadsheets for developing large accounting applications instead of using more reliable software products that are better suited to handling a large amount of data.

A computing sheet is a set of values linked by references, which constitute a sort of graph. The formula denotations of most spreadsheets, based on cells which refer to other cells, provide very low-level semantics for describing the spreadsheet structure. Mutatis mutandis (the necessary changes having been made), this is similar to the programming languages of the early 60s, with the **goto** statement used as the main means for structuring code. This analogy with programming languages is not fortuitous. Building a computing sheet is similar to building a program. For at least three decades, work in the programming world has been done in order to offer numerous tools and methods for building programs that are structured, reliable, extensible, etc. Apparently, such work has not yet been done in the area of spreadsheets or at least has had no visible effect in most software products.

Thus, the aim of Xxl is to provide a new spreadsheet model. In particular, it will result in the development of a unique language that will permit it to integrate mechanisms for cell typing, assertion definition and global descriptions directly within the spreadsheet. Cell typing will validate input or cut-and-paste values. Assertion mechanisms will prove the validity of the global spreadsheet description.

Documentation of spreadsheets is an important aspect; however, spreadsheet developers generally consider it superfluous. Actually, documentation tools are almost nonexistent in Xxl. Time must be taken to propose a mechanism to simplify and automate the documentation of cells and spreadsheets.

Programming is a complex activity requiring knowledge and savoir faire. Spreadsheets are very popular software products, used generally by non-specialists. For most of them, the use of a high-level programming language would be a true difficulty. It is necessary to provide them with a graphical, interactive interface, simple and user friendly, in order to describe the semantics of spreadsheets without having to use a programming language.

Conclusion

Xxl is an academic project developed by students at the University of Nice—Sophia Antipolis. It was born in 1996 to fill the need in the UNIX world for a user friendly, easy-to-use and public-domain graphical spreadsheet program. UNIX has a long tradition of open source software. In the early 80s, R. Stallman and his FSF paved the way. Today, it goes on in different ways with Linux and the Internet. During its development, Xxl took advantage of various free software products. In turn, its authors are happy to offer it free to the international community.

The latest version of Xxl provides a truly functional spreadsheet for small or mid-size needs. The model it defines is similar to that of all present-day

spreadsheets. However, this model is limited and needs to be entirely revised. Xxl is a stable experimental platform for future student projects to propose and define tools for a new spreadsheet generation.

Resources

Vincent Granet (vg@unice.fr) received his Ph.D. from the University of Nice—Sophia Antipolis in 1988. After lecturing in the University of Provence and in the University of Nancy I, he came back to the University of Nice—Sophia Antipolis in 1995 as an assistant professor in Computer Science. He is also a research staff member in the I3S laboratory (Computer science and signal processing) of the University of Nice—Sophia Antipolis, where he works in the “Languages” team.

Archive Index Issue Table of Contents

Advanced search

Network Programming with Perl

James Lee

Issue #60, April 1999

Using Perl to make network task is easy—here's how.

Perl has been called the glue that holds the Internet together because it is an extremely powerful text processing and CGI programming language. Although Perl was designed in the beginning to be a text manipulation language, it has evolved into a potent multi-purpose programming language. One area in which Perl has shown its power is that of network programming.

Perl makes network programming easy by providing built-in functions that can be used to create low-level client/server programs from scratch. Also, many modules are freely available that make programming common networking tasks simple and quick. These tasks include pinging remote machines, TELNET and FTP sessions. This article presents examples of each of these types of network programs.

Introduction

Client/server network programming requires a server running on one machine to serve one or more clients running on either the same machine or different machines. These different machines can be located anywhere on the network.

To create a server, simply perform the following steps using the built-in Perl function indicated:

- Create a socket with **socket**.
- Bind the socket to a port address with **bind**.
- Listen to the socket at the port address with **listen**.
- Accept client connections with **accept**.

Establishing a client is even easier:

- Create a socket with **socket**.
- Connect (the socket) to the remote machine with **connect**.

Several other required functions and variables are defined in the **Socket.pm** module. This module is probably already installed on your machine, but if not, it is available at the Comprehensive Perl Archive Network (CPAN), the official Perl source code repository (see Resources). To use this module in our programs, the following statement is required at the top of the program:

```
use Socket;
```

This statement will locate the file Socket.pm and import all of its exported functions and variables.

Viewing Module Documentation

All examples in this article use modules that are available at no cost from CPAN.

Perl modules are usually self-documenting. If the author of the module follows the generally accepted rules of creating a Perl module, they will add Plain Old Documentation (POD) to the module's .pm file. One way to view the POD for the Socket module (assuming Perl and Socket.pm were installed correctly) is to execute the following at the shell:

```
perldoc Socket
```

This command displays Socket.pm's POD converted to a man page. The output is a relatively thorough discussion of the functions and variables defined in this module.

Another way to view the documentation is to convert the POD to text using:

```
pod2text \
/usr/lib/perl5/i686-linux/5.00404/Socket.pm | more
```

The program **pod2text** is included in the Perl distribution, as are the programs **pod2html**, **pod2man**, **pod2usage** and **pod2latex**.

A Simple Server

Listing 1.

Our first programming example is a simple server running on one machine that can service only one client program at a time connecting from the same or a

different machine. Recall that the steps for creating a server were to create a socket, bind it to a port, listen at the port and accept client connections.

Listing 1, server1.pl, is the source code for this simple server. First, it is generally a good idea to compile using Perl's strict rules:

```
use strict;
```

This requires all variables to be declared with the **my** function before they are used. Using my may be inconvenient, but it can catch many common syntactically correct yet logically incorrect programming bugs.

The variable **$port** is assigned the first command-line argument or port 7890 as the default. When choosing a port for your server, pick one that is unused on your machine. Note that the only way to ensure you select a port that does not have a predefined use is to look at the appropriate RFC (see Resources).

Next, the socket is created using the socket function. A socket is like a file handle—it can be read from, written to or both. The function **setsockopt** is called to ensure that the port will be immediately reusable.

The **sockaddr_in** function obtains a port on the server. The argument **INADDR_ANY** chooses one of the server's virtual IP addresses. You could instead decide to bind only one of the virtual IP addresses by replacing **INADDR_ANY** with

```
inet_aton("192.168.1.1")
```

or

```
gethostbyname("server.onsight.com")
```

The bind function binds the socket to the port, i.e., plugs the socket into that port. Then, the listen function causes the server to begin listening at the port. The second argument to the listen function is the maximum queue length or the maximum number of pending client connections. The value **SOMAXCONN** is the maximum queue length for the machine being used.

Once the server begins listening at the port, it can accept client connections using the accept function. When the client is accepted, a new socket is created named **CLIENT** which can be used like a file handle. Reading from the socket reads the client's output and printing to the socket sends data to the client.

To read from a file handle or socket in Perl, wrap it in angle brackets (<*FH*>). To write to it, use the **print** function:

```
print SOCKET;
```

The return value of the accept function is the Internet address of the client in a packed format. The function sockaddr_in takes that format and returns the client's port number and the client's numeric Internet address in a packed format. The packed numeric Internet address can be converted to a text string representing the numeric IP using **inet_ntoa** (numeric to ASCII). To convert the packed numeric address to a host name, the function **gethostbyaddr** is used.

Let's assume all of the servers referred to in this article are started on the machine named server.onsight.com. To start the server on this machine, execute:

```
[james@server networking]$ server1.pl
SERVER started on port 7890
```

The server is now listening at port 7890 on server.onsight.com, waiting for clients to connect.

A Simple Client

Listing 2.

Listing 2, client1.pl, shows a simple client. The first command-line argument to this program is the host name to which it should connect, which defaults to server.onsight.com. The second command-line argument is the port number which defaults to 7890.

The host name and the port number are used to generate the port address using **inet_aton** (ASCII to numeric) and sockaddr_in. A socket is then created using socket and the client connects the socket to the port address using connect.

The **while** loop then reads the data the server sends to the client until the end-of-file is reached, printing this input to STDOUT. Then the socket is closed.

Let's assume all of the clients are started on the the machine named client.avue.com, although they could be executed from any machine on the network. To execute the client, type:

```
[james@client networking]$ client1.pl server.onsight.com
Hello from the server: Tue Oct 27 09:48:40 1998
```

The following is the standard output from the server:

```
got a connection from: client.avue.com [192.168.1.2]
```

Perl Makes Life Easy

Creating sockets using the functions described above is good when you want to control how the socket is created, the protocol to be used, etc. But using the functions above is too hard; I prefer the easy way—**IO::Socket**.

The module IO::Socket provides an easy way to create sockets which can then be used like file handles. If you don't have it installed on your machine, it can be found on CPAN. To see this module's POD, type:

```
perldoc IO::Socket
```

A Simple Server Using IO::Socket

Listing 3.

Listing 3, serverIO.pl, is a simple server using IO::Socket. A new **IO::Socket::INET** object is created using the **new** method. Note that the arguments to the method include the host name, port number, protocol, queue length and an option indicating we want this port to be immediately reusable. The new method returns a socket that is assigned to **$sock**. This socket can be used like a file handle—we can either read the client output from it, or write to it by sending data to the client.

A client connection is accepted using the accept method. Note the accept method returns the client socket when evaluated in scalar context:

```
$new_sock = $sock->accept()
```

and returns the client's socket and the client's IP address when evaluated in list context:

```
($new_sock, $client_addr) = $sock->accept()
```

The client address is computed and printed the same as in Listing 1, server1.pl. Then the socket associated with that client is read until end-of-file. The data read is printed to STDOUT. This example illustrates that the server can read from a client using < > around the socket variable.

A Simple Client Using IO::Socket

Listing 4.

Listing 4, clientIO.pl, is a simple client using **IO::Socket**. This time, a new object is created that connects to a host at a port using the TCP protocol. Ten strings are then printed to that server, then the socket is closed.

If the server in Listing 3, serverIO.pl, is executed and then the client Listing 4, clientIO.pl, connects, the output would be:

```
[james@server networking]$ serverIO.pl
```

```
got a connection from: client.avue.com [192.168.1.2]
hello, world: 1
hello, world: 2
hello, world: 3
hello, world: 4
hello, world: 5
hello, world: 6
hello, world: 7
hello, world: 8
hello, world: 9
hello, world: 10
```

Bidirectional Communication

It is possible to create servers and clients that communicate with one another in both directions. For instance, the client may send information to the server, then the server may send information back to the client. Therefore, network programs can be written so that the server and client follow some predetermined protocol.

Listing 5.

Listing 5, server2way.pl, shows how a simple server can be created to read a command from a client, then print out an appropriate response to the client. The module **Sys::Hostname** provides a function named **hostname** that returns the host name of the server. To insure output is seen as we print, IO buffering is turned off for the STDOUT file handle using the **autoflush** function. Then a **while** loop is executed that accepts connections. When a client connects, the server reads a line from the client, chopping off the newline character. Then a **switch** statement is executed. (The **switch** is cleverly disguised as a **foreach** loop, which happens to be one of my favorite ways of writing a **switch**.) Depending on the input entered by the client, the server outputs an appropriate response. All lines from the client are read until end-of-file.

Listing 6.

Listing 6, client2way.pl, shows the companion client. A connection to the server is made, then the client prints a few commands to the server reads the response and prints the response to **STDOUT**.

The following is the output of the client code in Listing 6:

```
[james@client networking]$ client2way.pl
server.onsight.com
Hi
server.onsight.com
```

```
Tue Oct 27 15:36:19 1998
DEFAULT
```

A Forking Client

If you want to write a client that accepts commands from STDIN and sends them to the server, the easiest solution is to write a client that forks a child. (A solution can be written using **select** that does not fork, but it is more complicated.) The client's parent process will read the commands from the user through STDIN and print them to the server. The client's child process will then read from the server and print the responses to STDOUT.

Listing 7.

Listing 7, clientfork.pl, is an example of a client that forks.

To fork in Perl, call the cleverly named **fork** function. It returns **undef** if the fork fails. If it succeeds, it returns **0** to the child, non-zero (the child's pid) to the parent. In clientfork.pl, an **if** statement checks the value of **$kid**, the return value from the fork. If **$kid** is true (non-zero, the child's pid), parent executes reading from STDIN printing to the server. If **$kid** is false (zero), the child executes reading from the server printing to STDOUT.

The following is the example session executing the client code in Listing 7, clientfork.pl connecting to the code in Listing 5, server2way.pl:

```
[james@client networking]$ clientfork.pl
server.onsight.com
NAME
server.onsight.com
DATE
Tue Oct 27 15:42:58 1998
HELP
DEFAULT
HELLO
Hi
```

When the parent process is finished reading from STDIN, it executes the **kill** function to kill the child process. It is very important the parent reap its child so that the child does not outlive the parent and become a zombie.

A Forking Server

Listing 8.

Servers usually don't handle only one client at a time. One approach to a server that can handle more than one client is a server that forks a child process to handle each client connection. Listing 8, serverfork.pl, is an example of a forking server.

One way for the parent process to reap its children is to define a subroutine and assign a reference to that subroutine to **$SIG{CHLD}**. (The hash **%SIG** is Perl's way of handling signals.) In this example, a subroutine named **REAP** is defined and a reference to this subroutine is assigned to **$SIG{CHLD}**. When the parent receives the **CHLD** (child terminated) signal, the REAP subroutine will be invoked.

Within the **while** loop that accepts all the client connections, the server forks. If the fork returns true, the parent is running and it executes the **next** statement which immediately transfers control to the **continue** block, performs the housecleaning step of closing the child socket and waits for the next client to connect. If the fork returns **undef**, then the fork failed, so the server dies. If the fork returns neither true nor **undef**, then the child is running, so the parent socket is closed and the child reads from the client and processes the client. When the child is finished processing the client, the child exits and is reaped by the parent.

Thread Programming in Perl5.005

Perl version 5.005 supports thread programming. This means a threaded networking program can be created to be either a server or a client.

Listings 9, 10, and 11 are three different versions of a client that logs into several web servers and determines the type of server being used (Apache, Netscape, etc).

Listing 9.

Listing 9, getservertype1.pl, shows a non-forking, non-threaded client. First, an array of hosts is created and initialized to a few web sites. The subroutine **doit** is defined to receive the web server name as an argument, open a client connection to that server at port 80 (the HTTP port), send the HTTP request and read each line of response. When the line starting **Server:** is read, it will extract the server name and store it in **$1**. Then the host name and web server name are printed. This subroutine is called for each host in the array of hosts.

Here is the output of getservertype1.pl:

```
processing www.ssc.com...
www.ssc.com: Stronghold/2.2 Apache/1.2.5 PHP/FI-2.0b12
processing www.linuxjournal.com...
www.linuxjournal.com: Stronghold/2.2 Apache/1.2.5 PHP/FI-2.0b12
processing www.perl.com...
www.perl.com: Apache/1.2.6 mod_perl/1.11
processing www.perl.org...
www.perl.org: Apache/1.2.5
processing www.nytimes.com...
www.nytimes.com: Netscape-Enterprise/2.01
processing www.onsight.com...
www.onsight.com: Netscape-Communications/1.12
```

```
processing www.avue.com...
www.avue.com: Netscape-Communications/1.12
```

Note that the hosts are processed in the same order as stored in **@hosts**.

Listing 10.

Listing 10, getservertype2.pl, is a forking version of getservertype1.pl. The forking occurs within the **foreach** loop. The fork is executed and if it returns true, the parent then executes the **next** statement to the next host name. If the fork returns **undef**, then the program dies. Otherwise, the child calls the doit function passing in the host, then exits. After the parent completes its work in the **while** loop, it waits for all child processes to finish, then exits.

Here is the output of getservertype2.pl:

```
processing www.ssc.com...
processing www.linuxjournal.com...
processing www.perl.com...
processing www.perl.org...
processing www.nytimes.com...
processing www.onsight.com...
processing www.avue.com...
www.onsight.com: Netscape-Communications/1.12
www.nytimes.com: Netscape-Enterprise/2.01
www.avue.com: Netscape-Communications/1.12
www.linuxjournal.com: Stronghold/2.2 Apache/1.2.5 PHP/FI-2.0b12
www.perl.com: Apache/1.2.6 mod_perl/1.11
www.ssc.com: Stronghold/2.2 Apache/1.2.5 PHP/FI-2.0b12
www.perl.org: Apache/1.2.5
Parent exiting...
```

Note that the hosts are not printed in the order stored in **@hosts**. They are printed in the order processed, the slower hosts taking longer than the faster ones.

Listing 11.

Listing 11, getservertype3.pl, is a threaded version. In the loop through the host names, a new **Thread** object is created. When creating the **Thread**, the new method is passed a reference to a subroutine that the thread will execute, as well as the arguments passed into that subroutine. The thread then executes its subroutine and when the subroutine returns, the thread is destroyed. Here is the output of getservertype3.pl:

```
processing www.ssc.com...
processing www.linuxjournal.com...
processing www.perl.com...
processing www.perl.org...
processing www.nytimes.com...
processing www.onsight.com...
processing www.avue.com...
www.nytimes.com: Netscape-Enterprise/2.01
www.onsight.com: Netscape-Communications/1.12
www.avue.com: Netscape-Communications/1.12
www.linuxjournal.com: Stronghold/2.2 Apache/1.2.5 PHP/FI-2.0b12
www.perl.com: Apache/1.2.6 mod_perl/1.11
```

```
www.ssc.com: Stronghold/2.2 Apache/1.2.5 PHP/FI-2.0b12
www.perl.org: Apache/1.2.5
```

Net::Ping Module

Listing 12.

The **Net::Ping** module makes pinging hosts easy. Listing 12, ping.pl, is a program similar to a program on my server that pings my ISP to keep my connection alive. First, a new **Net::Ping** object is created. The protocol chosen is **tcp** (the choices are **tcp**, **udp** and **icmp**; the default is **udp**). The second argument is the timeout (two seconds). Then an infinite loop is executed, pinging the desired host. The **ping()** method returns true if the host responds, false otherwise, and an appropriate message is printed. Then the program sleeps ten seconds and pings again.

An example output of Listing 12, ping.pl, is:

```
Success: Wed Nov  4 14:47:58 1998
Success: Wed Nov  4 14:48:08 1998
Success: Wed Nov  4 14:48:18 1998
Success: Wed Nov  4 14:48:28 1998
Success: Wed Nov  4 14:48:38 1998
Success: Wed Nov  4 14:48:48 1998
```

Net::Telnet Module

Listing 13.

The **Net::Telnet** module makes automating TELNET sessions easy. Listing 13, telnet.pl, is an example of connecting to a machine, sending a few system commands and printing the result.

First, a server and a user name are used. The user name defaults to the user running the script by assigning to **$user** the value **$ENV{USER}**. (The hash **%ENV** contains all of the environment variables the script inherits from the shell.)

Next, the password is requested, then read in. Note that turning off the stty echoing is done through a **system** call. It can also be done using the **Term::ReadKey** module.

Then, a **Net::Telnet** object is created. To log in to the server using this object, the **login** method is called. Several system commands are executed using the **cmd** method which returns the STDOUT of the system command which is then printed. Note that part of that output is the system prompt, which is printed along with the output of the command.

Also note that the code **$tn->cmd('/usr/bin/who')** is evaluated in list context and stored in **@who**, which is an array that contains all the lines of ouptut of that command, one line of output per array element.

After all of the system commands are executed, the TELNET session is closed.

Here is an example output of Listing 13, telnet.pl:

```
Enter password:
```

```
Hostname: server.onsight.com
[james@server james]
Here's who:
james    tty1     Oct 24 21:07
james    ttyp1    Oct 27 20:59 (:0.0)
james    ttyp2    Oct 24 21:11 (:0.0)
james    ttyp6    Oct 28 07:16 (:0.0)
james    ttyp8    Oct 28 19:02 (:0.0)
[james@server james]
What is your command: date
Thu Oct 29 14:39:57 EST 1998
[james@server james]
```

Net::FTP Module

Listing 14.

The **Net::FTP** module makes automating FTP sessions easy. Listing 14, ftp.pl, is an example of connecting and getting a file.

A **Net::FTP** object is created, the login is called to log in to the machine, the **cwd** changes the working directory and the **get** method gets the file. Then the session is terminated with **quit**.

There are methods to do many common FTP operations: **put**, **binary**, **rename**, **delete**, etc. To see a list of all the available methods, type:

```
perldoc Net::FTP
```

Here is an example output of Listing 14, ftp.pl:

```
[james@k2 networking]$ ftp.pl server.onsight.com
Enter your password:
Before
----------------------------------------
/bin/ls: *.gz: No such file or directory
----------------------------------------
After
----------------------------------------
perl5.005_51.tar.gz
----------------------------------------
```

Archive a Web Site

Using both Net::Telnet and Net::FTP, a very simple script can be created that can archive a directory structure on a remote machine.

Listing 15.

Listing 15, taritup.pl, is a Perl program that is similar to a program I use that logs in to my ISP and archives my web site.

The steps this program follows are:

- Start a session on the remote machine with TELNET.
- Go to the web page directory using **cd**.
- Archive the directory using **tar**.
- Start an FTP session to the remote machine.
- Change to the directory containing the tar file.
- Get the tar file.
- Quit the FTP session.
- Back in the TELNET session, delete the tar file on the remote machine.
- Close the TELNET session.

This program outputs text to let the user know how the script is progressing.

Summary

Perl is a powerful, easy-to-use programming language. That power and ease of use includes network programming due to many built-in functions and modules. Best of all, it's free.

Resources

James Lee is the president and founder of Onsight (http://www.onsight.com/). When he is not teaching Perl classes or writing Perl code on his Linux machine, he likes to spend time with his kids, root for the Northwestern Wildcats (it was a long season), and daydream about his next climbing trip. He also likes to receive e-mail at james@onsight.com.

LINUX JOURNAL

Linux in Enterprise Network Management

Leo Lahteenmaki

Issue #60, April 1999

Providing network information to customers on an Intranet saves both time and money for this international chemical company.

Kemira is a chemical company which employs approximately 10,000 people and has plants in Europe, the U.S. and Asia. Kemira got into IP-networking early by building KemNet, an IP-based Intranet out of Cisco routers, in 1989.

Mission-critical information systems, such as process automation computers, are located on-site in many of Kemira's plants. IT professionals run and manage these systems and they rely on KemNet to do their job efficiently. These IT professionals are the primary internal customers for my unit, and we support them by determining answers to the following questions:

- How is the network performing?
- Do they have too little or too much network capacity?
- Is the telecommunications operator delivering as promised?
- Is a particular problem network-related or not?
- If KemNet is the cause of the problem, how will we avoid it in the future?

The challenge for our department is to provide our IT professionals with enough information so they can handle most network management tasks on-site. This allows us to keep our organization small (currently two employees each working half-time) and costs down.

Since the people using network management services are located on three continents, we decided to use e-mail and web technology to create a “self-service network management center” in Kemira's Intranet.

Many methods can be used to extract information from an IP router network. One nice feature of Cisco routers is their ability to log all types of network incidents using a **syslogd** daemon. These syslog messages can then be further processed with Perl scripts to create web pages or send e-mail messages. These messages are a great help in finding problems in routers and links or even in measuring usage of dial-in links.

Another popular method of getting network information is by using SNMP (Simple Network Management Protocol), the standard for managing IP networks. Many open-source tools for gathering and processing SNMP data are available. One tool we found useful and easy to set up is MRTG (Multi Router Traffic Grapher), which gathers traffic load information from router interfaces. (See Figure 1.)

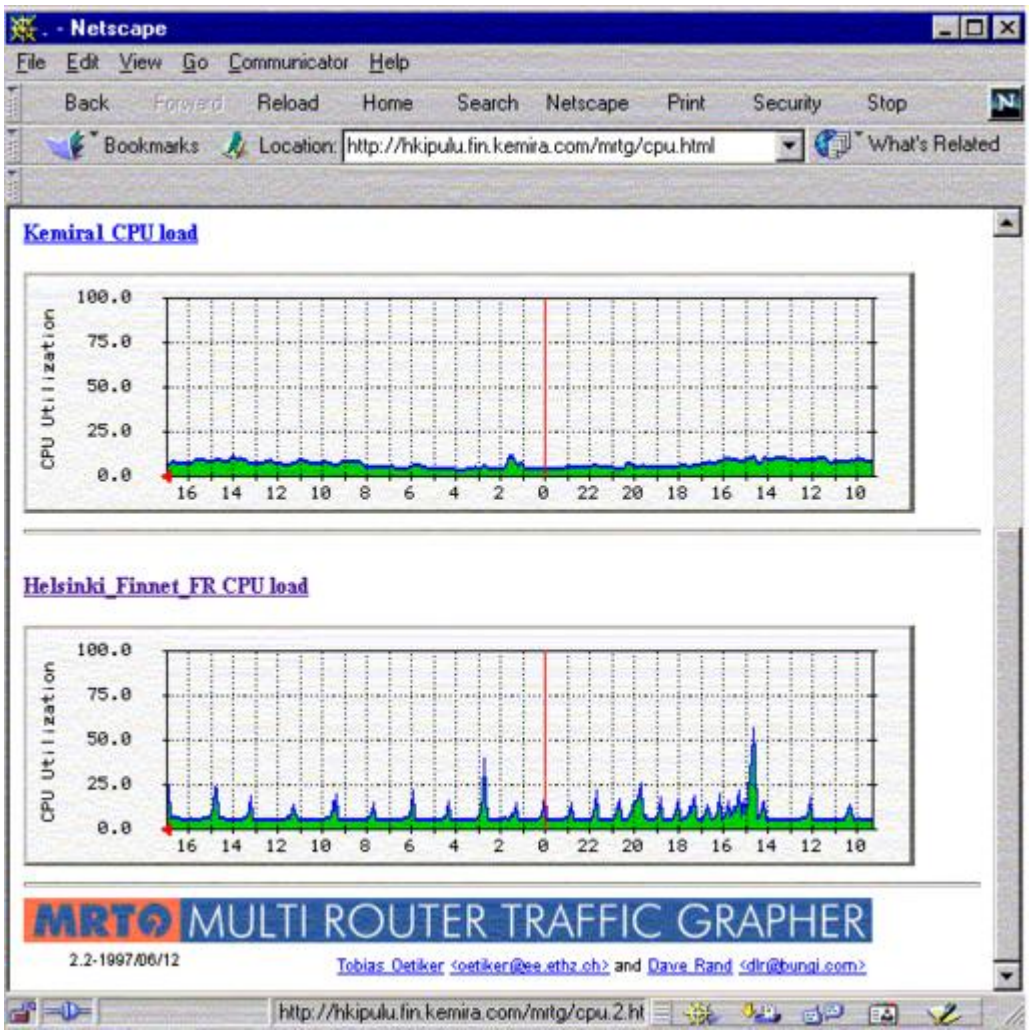


Figure 1. MRTG Screenshot

A third tool for getting performance information is good old **ping**. By periodically measuring ping round-trip times, you can find the times when your

network is most congested. Then, by checking the actual traffic volume using mrtg, you can see if the telecom operator is delivering what he has promised.

Most international data communication lines are delivered using frame-relay technology. Frame-relay services are priced using a committed information rate (CIR) for all conditions and access speeds. Measuring sometimes gives interesting results and may save your company money in the next frame-relay agreement.

With a 100MHz Pentium Linux server, a couple of Perl scripts, Apache and a few open-source tools, we turned these three network information sources into an automated, web-based, network information center. Our customers can now troubleshoot those problems they believe to be KemNet-related much faster, saving working hours and downtime.

Another application for Linux is work-flow automation. Our trouble-ticket volume is so small that it did not justify buying a full-blown trouble-ticket system. Instead, we use Hypernews, a web-based news group collaboration tool.

Linux in the Corporate World

Maintainability is always a question when using open-source products. For us, the question is easy because our network management services would not exist without open-source tools. We tried to deliver “self-service network management” using a well-known and expensive commercial product. We soon found that even keeping this tool running required frequent visits from a vendor's consultant with a fee of over $1000/day. Using it to make the network information available on the Web would have been extremely expensive to set up and maintain.

All our self-built software consists of short (a maximum of 20 lines) Perl scripts run from **cron** or CGI. Anyone with minimal Perl experience will be able to maintain these scripts. We don't use compiled languages and keep our scripts as simple as possible to ease maintenance.

Security is another issue that needs constant attention. In security audits, I have noticed consultants give extra attention to each Linux box found in the corporate Intranet. Fast-developing Linux software may introduce new security problems, and the powerful features of Linux give both the cracker and the administrator the potential to generate security holes.

Summary

Our customers feel they can do their job better now that they can access real-time network information with their browser. A telephone call to a human operator sitting in front of his expensive network management workstation cannot give them this service. We also feel that open-source tools and in-house scripts can be easily maintained and secured for a corporate environment.

We are expanding our use of the Linux platform into new applications such as directory services, log file management and network traffic analysis.

Tools for Linux-Based Network Management

Leo Lahteenmaki (leo.lahteenmaki@kemira.com) started playing with Linux when the first IP-stack became available. When not working, Leo likes to ski, play tennis or fish.

Archive Index Issue Table of Contents

Advanced search

Alphabet Soup: The Internationalization of Linux, Part 2

Stephen Turnbull

Issue #60, April 1999

Mr. Turnbull takes a look at the problems faced with different character sets and the need for standardization.

A large body of standards has evolved to handle the problems with text manipulation I presented last month. In general, ad hoc handling methods are considered to be localization, while a method that conforms to some standard and is generalizable to many cultural environments is considered internationalization.

POSIX

Currently, the central standard for internationalization is the locale model of POSIX. Unfortunately, in the current state of the art, localization via the POSIX model is something of a Procrustean bed. For example, in Japanese there are two common ways of notating the currency unit yen: postfixing ¥ and prefixing &yen. It is not uncommon for both conventions to be used in the same document in different contexts: the former is common in running text, the latter in tables. POSIX does not provide for this. It is easy enough to implement by creating a Japanese-table locale to complement the Japanese-text locale, but this places the burden of setting the correct locale on the application programmer. Although much smaller in scale, this burden is much like that imposed by multilingualization. Nor was POSIX designed to support such fine discrimination; this is better left to the individual application anyway.

POSIX-style internationalization provides a comfortable, functional environment for almost all users and applications. Specifically, a POSIX locale determines:

- the character set and encoding to be used
- classification of characters (e.g., alpha, hex-digit, whitespace, etc.)
- the sorting order for strings in the language

- digit separator and decimal-point conventions
- date and time presentation
- currency presentation
- message format (in particular, strings for *yes* and *no*)

All of these features are implemented by changing the functionality of standard library functions or by adding new ones. That is, the **isalpha** function in libc no longer consults a fixed table, but instead the table is varied according to the current locale. Displaying monetary values can be done by using the new function **strfmon**. Unfortunately, the locale support in Linux libc is still only partially documented as of libc-2.0.7t; no man page for strfmon exists, although there is an entry point in the library. A useful discussion by Ulrich Drepper, one of the authors of GNU libc, may be found at http://i44s11.info.uni-karlsruhe.de/~drepper/conf96/paper.html.

POSIX Internationalization Levels

The POSIX standard defines a number of levels of compliance with internationalization standards. These levels are a somewhat useful guide to how far an internationalization effort has progressed. Level 1 compliance is achieved when a system is 8-bit clean. Obviously this is a bare minimum, since some characters may get corrupted. Level 2 compliance is achieved when a flexible system for producing localized time, date and monetary formats is implemented. As described above, these facilities are provided by GNU libc, so disciplined use of appropriate formatting functions and the **setlocale** call is going to be sufficient for most applications to achieve Level 2 compliance. Level 3 compliance is achieved when the application can use localized message catalogs. This facility is provided by the GNU **gettext** library. Controlling gettext is nearly as simple as setting the locale. Unfortunately, the rules of precedence are somewhat different. However, disciplined use of gettext and its supporting functions will make localization much easier. (See “Internationalizing Messages in Linux Programs” by Pancrazio de Mauro, March 1999.)

Level 4 refers to Asian language support. The Asian languages are given a special status because of the variety of complex subsystems needed to support them. For example, many implementations of X have two separate families of string display functions, one for strings encoded in one byte and another for strings composed of characters encoded in two bytes. In Japanese, one- and two-byte characters are mixed freely, so an internationalized application which needs to deal with Japanese would have to analyze strings into one-byte and two-byte substrings on the fly. In fact, dealing with Japanese by itself forces the programmer to deal with many of the problems posed by true multilingual applications.

Character Sets

It would be nice if we could think of character sets as corresponding directly to the scripts we use to write by hand, but unfortunately things are not so simple. For example, there are nearly 200 countries in the world, each with its own currency. Of course, many share the same symbol, but it is clear that if your keyboard had a key for every currency symbol, it would be about twice as big as the one you use today. Other useful symbols are the paragraph and sectioning marks used by lawyers and the various operators and non-Latin symbols used by mathematicians. Since new characters are being created all the time (for example, the symbol for the new European monetary unit, the euro), it is impossible to include them all. So in fact, a character set is someone's idea of a useful set of characters.

Representation as bit strings inside a computer imposes further constraints. Since modern computers all work in terms of bytes as the smallest efficient unit of access, there is a big difference in the space and processing requirements for text based on a 256-character set, which can be encoded in a single byte, and a 257-character set, which cannot. One might think that the extension to two bytes, or 65536 characters, would be enough to satisfy anyone, but it turns out that even this is not enough. The process of selecting about 20,000 ideographic characters of Chinese origin occasioned many arguments while the Unicode character set was being designed. Even those 20,000 may not be enough; while there are only a few people in the world who care about some of the excluded ideographic characters, to them it may be the most important character in the world, as it is the one they use to write their name.

The result is that many character sets have been designed and populated, and standards have been written to codify their use.

ASCII

The most influential standard of all is the American Standard Code for Information Interchange, abbreviated ASCII. This is a list of the 128 7-bit bit strings, with an assignment of each one to either a character commonly used in American English or a control function. Many of the control functions are not used today, but so much software has been written on the assumption that hex values 0x00 to 0x1F are not printing characters that no one considers assigning a few more characters to some of those code points.

Because nearly all existing computer languages are compatible with the ASCII character set, ASCII in some form is a subset of most electronic character sets. However, there are many variants. For example, the JIS Roman character set used in most Japanese computers is almost identical to ASCII, except that a couple of the glyphs are changed and the Japanese yen symbol is substituted

for the backslash. In order to codify this development, ASCII-like character sets are defined by the International Standards Organization (ISO) in standard ISO 646. U.S. ASCII is designated the international reference version for ISO 646 and is occasionally referred to as ISO 646-IRV (for example, in naming fonts for the X Window System).

The ISO 8859 Family of Character Sets

ASCII is simply not sufficient for use in an internationalized environment. For example, most European languages use accented characters. Certainly, it is possible to represent “Latin small letter a with acute accent” (á) as a two character ligature (e.g., **'a**), but this is inconvenient for sorting and possibly ambiguous. Furthermore, it is not obvious how to represent the caron using only ASCII characters. In order to maintain compatibility with ASCII for the sake of existing software, and accommodate many of the countries most intensively using computers, the ISO 8859 standard was designed. ISO 8859 had three main goals: maintain ASCII compatibility, implement within the constraints of ISO 2022 and provide the broadest coverage of languages within a single-octet encoding. Unfortunately, these three goals are not compatible. Several important scripts which can be encoded in a single octet require several dozen code points each for their characters because they do not overlap with ASCII or each other: Greek, Russian, Hebrew and Arabic.

The solution arrived at was not to define a single character set, but rather a family of character sets. Each ISO 8859 character set contains ASCII (ISO-646-IRV) as a subset, and the encoding is defined so that, interpreted as integers (C chars), the ASCII characters are encoded identically in ASCII and ISO 8859. Then a list of supplementary character sets, each including at most 96 characters to conform to ISO 2022 (described below), was defined. These supplementary characters are then assigned to the code points 0xA0 to 0xFF. Where the supplementary set is derived from an alphabet, the natural collating order is followed, but for the collections of accented characters the order is necessarily arbitrary. The current supplementary character sets are listed in Table 1.

Table 1.

Unicode and the ISO-10646 Universal Character Sets

The next step is to unify all of the various character sets. Of course, the national standards have two main advantages. They are space-efficient, encoding the characters needed for daily use and computer programming in one byte, and they are time-efficient, since they can be arranged in the natural collating order. The second advantage has already been conceded by the majority of European languages when using encodings in the ISO 8859 family. Indeed, ISO 8859-1 has been an enormous success since it effectively unifies all the major Western

European and American languages in a single multilingual encoding. With system library support for sorting (the LC_COLLATE portion of POSIX locales), it is hard to justify using anything else where it will serve.

In this context, it was natural to try to extend the success of ISO 8859 by abandoning the efficiency of one-byte encodings in favor of a single comprehensive encoding for all characters used by all the world's languages. Two complementary efforts, proceeding in parallel, were conducted by a commercial consortium and the ISO. Unsurprisingly, the ISO's working group called its effort by the ponderous name Universal Multiple-Octet Coded Character Set (abbreviated UCS), while the commercial consortium adopted the sprightly “Unicode”. Also unsurprisingly, the Unicode Consortium (driven by the commercial advantages of a uniform two-byte encoding) was able to formulate a standard unifying nearly all of the world's scripts in a single two-byte encoding by 1991, as well as codifying a dictionary of properties of each character guiding such usages as ligatures and bidirectional text, while the ISO ended up defining both a two-byte version and a four-byte (31-bit) version of the UCS in 1993 without the additional properties. Also in 1993, the Unicode and UCS-2 character sets and encodings were unified, although each standard retains unique features.

Why separate efforts? Surely 65336 different characters are enough for anyone. Who needs two billion characters?

The reason for separate efforts is easy enough to explain. The Unicode effort was driven by the commercial advantages of a single encoding. Much effort has been expended in the standardization of Internet protocols, first working around the problems caused by “8-bit-dirty” Internet software, then in adding support for Asian languages, and finally in creating protocols for negotiating character sets. It would be nice if all that effort and the necessary implementation inefficiencies could be avoided by having one standard encoding. As we will see, it is not that easy, but standardizing on Unicode could result in large cost savings, both in development and processor time and protocol overhead.

On the other hand, the ISO group was primarily concerned that a truly universal framework be created so as to avoid the need for yet another “universal” standardization effort in the future. It worried more about generality and eschewed standardizing poorly-understood areas, such as treatment of bidirectional text. In fact, UCS-4 currently contains only those characters defined by the Unicode standard, adopted en masse as the Basic Multilingual Plane of UCS-4 and equivalent to UCS-2.

The reason for their concern is it is already painfully obvious 65536 characters are *not* enough for some purposes. Although over 18,000 unassigned code positions remain in Unicode, classical scholars of hieroglyphics or Chinese could rapidly fill these positions with ideographs. The current set of “unified Han” (Chinese ideographs used in Chinese, Japanese, Korean and Vietnamese) was reduced to 20,902 only through a highly contentious unification process, suggesting that some of the controversial characters might be reassigned to code points. Archaic Hangul (composed Korean syllables) would add thousands more. Unicode also explicitly excludes standardized graphic notations such as those used in music, dance and electronics. It is clear that a truly universal character set will easily exceed the limit of 65536 imposed by a two-octet encoding.

Why does ISO 10646 specify a 31-bit encoding? Current hardware is byte-oriented, but there is no particular reason to stop at 24 bits, since only certain video hardware can efficiently use three-byte words of memory. The word size most efficiently accessed by most current hardware is four bytes. With potentially billions of characters, it was considered wise to reserve a bit in each character for arbitrary internal processing purposes; however, this bit must be cleared before passing the character on to an entity expecting a UCS character.

Similarly, large contiguous private spaces have been reserved containing 1/8 of the three-octet codes, i.e., those with the high octet 0, and 1/4 of the four-octet codes. This means that an application can embed entire national standard character sets in this space in a natural way (in particular, preserving their orderings) if desired, without any possibility of conflict with the standard, current or any future extensions. ISO 10646 does not necessarily recommend such techniques, but certainly permits them. This still leaves over 1.5 billion code points reserved for future standardization; it seems certain most will remain reserved but unused for a good long time.

However, it seems unlikely that Unicode, let alone UCS-4, will soon have the success enjoyed by ISO 8859-1. First, the Oriental languages' digital character set standards are not yet satisfactory, in part because the languages are not fully standardized. Standardization efforts for all the Han character languages remain active. If the Japanese, for example, have not yet settled on a national character set, how can they be satisfied with the unified Han characters of Unicode? A recent tract entitled *Japanese is in Danger!* claims that Unicode will be the death of the Japanese language, and many computer-literate Japanese show varying degrees of sympathy with its arguments.

Second, in multilingual texts it may be desirable to search for some specifically Chinese character (as opposed to its Korean or Japanese cognates). In Unicode, this requires maintaining substantial amounts of surrounding context which

would contain markup tags indicating language and would be impossible by definition in Unicode-encoded plain text. Although you could point to similar difficulties with ISO 8859-1 text, it is not the same. A Chinese character is a semantic unit with specific meaning, unlike an alphabetic character. In fact, the Han unification process normally ignores semantics. Thus, it confounds a Japanese character with the same shape as a given Chinese character, but a different meaning. ISO 8859-1 characters, on the other hand, are rarely searched for in isolation; if so, they have no semantic content.

Third, Asians are simply not yet as multilingual across the Asian languages as Europeans are across European ones, although this is changing rapidly. Still, it is unlikely that we will ever see an “Asian Switzerland” with Chinese, Japanese and Vietnamese simultaneously in use as official languages. Thus, the advantage of Unicode over national standards is not so great.

Fourth, from the Western European point of view, most of the gains to a single character set supporting multilingual processing have already been achieved by ISO 8859-1. Western Europeans have little need for Unicode.

In the near future, Unicode will be most useful to computer and operating system vendors, including Linux. By supporting Unicode as the basic internal code set, an unambiguous way is provided to avoid linguistic confusion. Adding new languages will simply be a matter of providing fonts, a Unicode-to-font-encoding mapping table and translating the messages. No additional programming effort will be necessary, and backwards compatibility is guaranteed. This is not trivial. An example is given below of a kernel patch used to make directory listings of Japanese Windows file systems mounted with either the MS-DOS or VFAT file systems readable. This kernel patch is certainly never going to be integrated into the kernel source code, because it is impossible to ensure it won't mess up non-Japanese names.

National Standard and Private Character Sets

Besides the national standard character sets mentioned above, many others are still in common use. A few of the more important ones include Russian's KOI-8 (alternative to ISO 8859-5), ISCII for Indian languages written in the Devanagari script and VISCII for Vietnamese. Of course, U.S. ASCII is available.

Other important character sets are those defined by industry or individual firms. An important characteristic of these private character sets is that their encodings often do not conform to the ISO 2022 standard, making interchange among systems difficult. Microsoft is at the forefront, defining and registering myriads of Windows character sets. Most of these are ASCII derivatives and closely related to ISO 8859 encodings, so although the small differences are annoying to programmers, they are often insignificant to users. However, in the

field of Asian languages the non-ISO-2022-compliant encodings called Shift JIS, an encoding for Japanese used by Microsoft and Apple operating systems, and Big 5, an encoding for traditional Chinese defined by a consortium of five large Taiwanese manufacturers, are important. In both cases, portions of the code space not used by international standard character sets were employed.

In the case of Shift JIS, the idea was to include the so-called half-width katakana, the 70 or so characters necessary to phonetically transcribe the Japanese language. Rarely used in normal text, they are somewhat convenient for file names in the DOS 8.3 format. This requires that they be encoded as a single octet. The Japanese standard JIS X 0201 encodes ASCII in its usual code points and places the katakana in the octets with values 0xA1 to 0xDF. Shift JIS is based on this standard and uses a simple algorithm to transform standard JIS kanji codes into two-octet codes with the first octet in the ranges 0x81 to 0x9F and 0xE0 to 0xEF, which are unused by JIS X 0201.

Character Set Extension: ISO 2022

Unicode and UCS-4 solve the problem of character representation permanently. However, as seen above, Unicode is not quite sufficient for all purposes and UCS-4 is much too wasteful for general use. Furthermore, an enormous amount of hardware and software is oriented toward one-octet character sets. Thus, the ISO 2022:1994 standard for code extension techniques (in particular, using several character sets in one data stream), the most recent edition of a standard first published as ECMA-35 by the European Computer Manufacturer's Association in 1971, remains relevant.

ISO 2022 is a rather abstract standard. A brief outline of its provisions follows.

- Division of codes into 7-bit and 8-bit types; the 256 code points in the 8-bit table are divided into the left (L, 0x00 to 0x7F) and right (R, 0x80 to 0xFF) halves. 7-bit codes are considered to use only the left half.
- Further division of the 128 points in each half into control (C, 0x00 to 0x1F) and graphic (G, 0x20 to 0x7F) codes.
- Codes 0x1B (escape), 0x20 (space) and 0x7F (delete) in CL and GL are fixed. Codes 0xA0 and 0xFF in GR are often left unused.
- Provisions are made for handling control characters similar to those for graphic characters described below, but these are uninteresting in a discussion of internationalization.
- Graphic character sets must be encoded in a fixed number of bytes per character. Either all bytes of all characters are in the range 0x20 to 0x7F, or all bytes of all characters are in the range 0xA0 to 0xFF. A character set in which the bytes 0x20 and 0x7F or 0xA0 and 0xFF are *never* used is

referred to as a 94*n*-character set, where *n* is the number of bytes. Otherwise, the character set is a 96*n*-character set.

- An encoding may use up to four character sets simultaneously, denoted G0, G1, G2 and G3. G0 must be a 94*n*-character set; the other three may be 96*n*-character sets. The interpretation of a byte depends on the shift state. Any of G0, G1, G2 or G3 may be invoked into GL by the *locking shift* control codes LS0, LS1, LS2 and LS3 respectively. When the character set G0 is shifted into GL, then G0 is used to interpret bytes in the range 0x20 to 0x7F. Similarly, in 8-bit encodings, *right locking shifts* are used to invoke character sets G1, G2 or G3 into GR by the control codes LS1R, LS2R and LS3R. Then that character set is used to interpret bytes in the range 0xA0 to 0xFF.
- A single character may be invoked from the G2 and G3 sets by use of the *single shift* control codes SS2 and SS3.
- Escape sequences are provided for the purpose of *designating* new character sets into the G0, G1, G2 and G3 elements.

A given version of ISO 2022 need not provide all of the above shifting and designating facilities. ASCII, for example, provides none. To the extent that they are provided by a derivative standard, the control codes must take the values as shown in Table 2.

Table 2.

Three examples of codes which may be considered applications of the ISO 2022 standard are ASCII, ISO 8859-1 and EUC-JP. ASCII is the standard encoding for American English. It is a 7-bit code with the ASCII control codes designated to C0, the ASCII graphic characters designated to G0, and C1, G1, G2 and G3 not used. C0 is invoked in CL; G0 is invoked in GL. No shift functions are used.

ISO 8859-1 is an 8-bit code, with C0 left unspecified (but normally C0 has the ASCII control characters in it), the ASCII graphic characters are designated to G0 and the Latin-1 character set is designated to G1. C1, G2 and G3 are unused. C0 is invoked in CL and G0 is invoked in GL. No shift functions are used.

Packed-format EUC-JP is an 8-bit code, with C0 unspecified but normally using the ASCII control characters; the JIS X 0201 Roman version of ISO 646 designated to G0; the main Japanese character set JIS X 0208 containing several alphabets, punctuation, the Japanese kana syllabaries, some dingbats and about 6000 of the most common kanji (ideographs) designated to G1; the half-width katakana from JIS X 0201 designated to G2; and the JIS X 0212 set of about 8000 less common kanji designated to G3. C0 is invoked in CL, G0 is invoked in GL and G1 is invoked in GR. No locking shift functions are used. Half-

width katakana and the JIS X 0212 kanji must be accessed using the single shifts SS2 and SS3 respectively, and they are shifted into GR.

Finally, ISO 2022 is commonly used in Internet mail and multilingual documents. The 7-bit version is used and every character set must be designated to G0 before use.

The single most important aspect of ISO 2022 is that code points in the range of ASCII control characters may not be used for graphic characters. This means that text files using encodings conforming to ISO 2022 will behave like text (with line breaks and not causing strange behaviour in your terminal or emulator) when displayed. If you do not have the fonts or your software does not understand the designation escape sequences, you will see gibberish, but at least your terminal will continue working.

A second useful fact is that in most cases ASCII or some version of ISO 646 is designated to G0. An encoding like EUC-JP with ASCII designated to G0 and invoked to GL and all of the other character sets invoked to GR is “file system safe” in 8-bit clean systems. This is more or less the definition of the EUC variant of ISO 2022.

Some encodings which do not conform and thus often cause problems in software not specifically prepared for them are Shift JIS, Big 5, VISCII and KOI-8. Shift JIS in particular annoys me, because I dual-boot Linux and Windows 95 for Japanese OCR, conversion of Microsoft Word documents to plaintext and FreeCell, and directory listings with Japanese in them are invariably messed up. Fortunately, I find yet to have a reason to try to access a file named in Japanese. Kernel patches are available which help deal with this, but they are unofficial and will stay that way because they are inherently dangerous. That is, they work with Japanese most but not all of the time, and they will not handle non-Japanese 8-bit encodings correctly.

Internet Messaging

One of the earliest and most important applications for the Internet is messaging, either direct to recipients (electronic mail) or broadcast (Usenet newsgroups). From the internationalization point of view, these are basically the same; internationalization doesn't care about the transmission mechanism, only how the content is handled.

Because messaging was an early application, it assumes a rather restricted environment. In particular, it assumes the data stream is limited to 7-bit bit-strings, and one cannot even be sure that all ASCII characters will be transmitted without error. In particular, if a message originates in the UNIX world, is passed through BITNET, i.e., EBCDIC encoding and back to UNIX, some

characters are likely to be corrupted. Of course, these days such corruption is unlikely, but when the standards were designed, it was commonplace. Now these restrictions are defined in standards and widely implemented in software, so they are likely to continue for the foreseeable future, even though the hardware and software for Internet transmission of data is extremely reliable.

The Internet mail transmission protocol (SMTP) is defined in RFC-821. The main provision of interest is that the transmission channel must transmit all 128 ASCII characters properly. 8-bit-clean channels are encouraged, but implicitly 7-bit characters are the norm. Internet messages are standardized in RFC-822 for electronic mail and RFC-1036 for Usenet. RFC-1036 adopts RFC-822 nearly in full, so I will refer to these three standards together as RFC-822.

RFC-822 is intended first of all to be compatible with RFC-821. The content of a message is divided into the part that is relevant to the mail transport system, the *headers*, and the part that is irrelevant to transporting the message, the *body*. RFC-822 allows users to send 8-bit content in the body at their own risk, but the headers must be in a 7-bit code, in particular, ASCII. This is rather annoying to non-English-speaking users. To permit non-English text in subject headers and in comments (particularly full names associated with addresses) and to provide reliable transport for non-ASCII body content, both non-English text and binary data of various kinds, the Multipurpose Internet Mail Extension suite of protocols was defined. Today, this standard occupies no less than five RFCs (2045-2049). We will be interested only in those parts related to internationalization.

MIME Transfer Encodings

The MIME transfer encodings are like the UCS transformation formats discussed above. They allow arbitrary content to be expressed in a way that will not choke the transmission channel or be damaged by it. MIME defines two transfer encodings, *quoted printable* and *BASE64*.

The quoted-printable encoding is very simple. Any octet may be represented by its hexadecimal code, preceded by an equals sign. So a space character is represented as =20 and the Spanish small enye (ñ) is =F1. The Latin capital letter A is =41. However, in general these are used only in three circumstances. First, since the equal sign is an escape character, it must be represented by =3D. Second, some software strips trailing whitespace, in particular on systems with record-oriented storage that do not use control characters to represent line breaks. A space or tab that ends a line will be encoded =20 or =09, respectively. This is important to the signature convention used on Usenet newsgroups. Finally, non-ASCII octets including most control characters will be

encoded. Thus, the quoted-printable encoding is intended for applications, such as Western European languages, where most characters come from the basic Latin (i.e., ASCII) set. In fact, one quickly learns to accurately read quoted printable text without decoding it.

Note that this is a transfer encoding. It is a purely mechanical transformation and provides no information about the intended meaning of the character. Although ñ is one interpretation of =F1, there are many others including a different one for each of the ten ISO 8859 character sets. Quoted printable encoding provides no indication of which is intended.

The BASE64 encoding is intended to be a robust encoding for arbitrary binary data, including images and audio. However, it is also commonly used for languages like Japanese where interpreting each octet separately as an ASCII character is illegible without decoding. It is more efficient than quoted printable, using only 33% more space than the original text, where each quoted character uses three times as much space as the unencoded octet. BASE64 is similar to the famous **uuencode** format long used in UNIX for the same purpose, but the characters used for the encoding are limited to the 52 Latin letters, the 10 decimal digits, the plus sign and slash.

The equals sign is also used, as padding. The reason for this choice is that base 64 is a convenient radix for byte-oriented encoding, since four base-64 digits can encode 24 bits or 3 octets. The characters chosen are passed intact by all known systems, which is not true of some of the punctuation marks used in the **uuencode** algorithm. The encoding algorithm is obvious:

1. Break up the data stream into groups of three octets. The last group may have one or two octets and will be treated specially.
2. For each group of three, concatenate the octets into a 24-bit string, then break it into four 6-bit groups. Interpret each as a 6-bit binary integer and index into the table above. This results in a group of four base-64 digits. Add them to the output.
3. If there is a remaining group, it has either one or two octets in it. Add one or two null octets to complete the group of three. Now treat it as in Step 2, except that if there was one octet in the group, add the first two base-64 digits to the output and pad the end with two equals signs to make a group of four. If there were two octets in the final group, add the first three base-64 digits to the output and pad with a final equals sign to make a group of four.

Notice that by using the equals sign it is always possible to exactly decode the original text; there will not even be a spurious null character at the end.

Furthermore, the algorithm is very fast and space-efficient, given the restrictions.

MIME-specific Headers

A message conforming to the MIME standard must have a version header of the form

```
MIME-Version: 1.0
```

Some mailers are sufficiently picky as to refuse to do MIME processing on mail lacking a valid MIME-Version header. This would be amusing, except for the fact that many mailers either do not implement the MIME functions correctly, produce an illegal MIME-Version header, or fail to insert the MIME-Version header at all.

The only version of the MIME header formats is 1.0. The MIME standard has undergone several revisions and expansions, but the basic format has remained unchanged at version 1.0. These revisions and standards have added new values for some of the parameters and specified interpretations for some ambiguous areas, but the syntax is unchanged. Case is irrelevant, in both the header tags and the values. The style of capitalization used below is more or less conventional, but not required.

One way to protect the content, or at least check that it has not been truncated, is to provide a Content-Length header. This is allowed by the MIME standard. The general type of encoding of the body is stated in the content-transfer-encoding header. The default is

```
Content-Transfer-Encoding: 7-bit
```

Other allowed values are “quoted-printable” and “base64” (both implicitly 7-bit) and “8-bit”.

Next, the content type of the body is specified. In most messages it will be plain text, specified as

```
Content-Type: text/plain
```

Other text types commonly found in mail these days are **text/rich** and **text/html**. A forwarded message (with no prefatory comments) may have content type specified as **message/rfc-822**. Messages can also be multipart. This is commonly used to add multimedia attachments, but can also be used to break up the body into components in different languages.

The MIME standard specifies that the character set is ASCII unless otherwise noted. RFC-822 requires that all headers be ASCII, so the MIME character set specification applies only to the body of the message. This specification is done using the **charset** parameter of the content type header. The default could be explicitly specified as

```
Content-Type: text/plain;charset=us-ascii
```

Note that the optional parameters are specified in **keyword=*value*** form. The correct way to specify ASCII is "us-ascii", because that is the preferred form as registered with the IANA. A list of valid character sets for MIME is at http://www.hunnysoft.com/mime/. Europeans will commonly use

```
Content-Transfer-Encoding: 8-bit
Content-Type: text/plain;charset=iso-8859-1
```

The Japanese standard for electronic messages is a version of ISO 2022 called ISO-2022-JP. In fact, this encoding needs to be extended only slightly. It can be used for Chinese and Korean as well and even as a multilingual encoding. The extended version is known as ISO-2022-JP-2 or ISO-2022-INT.

MIME-encoded Words: Non-ASCII Text in Headers

The MIME standard also provides a mechanism for putting non-ASCII text in headers. RFC-822 makes this illegal, so use of this mechanism will result in gibberish being displayed by mail programs that do not implement MIME. However, most mail programs today are MIME-aware, so this should not present any problems. If your correspondents complain, tell them to get a MIME-aware mailer.

The mechanism is simple. Non-ASCII text is encoded using either quoted-printable encoding or BASE64 encoding according to convenience, and bundled up into an *encoded word*. The reason it must be bundled into an encoded word is that the Content-Type header applies to the body, and if the body is multipart, there will be no **charset** parameter. Using a special header to control the format of headers seems silly, so the encoded word itself will contain the necessary character set information.

The format of an encoded word begins with the characters =?, continues with the name of the encoding used, the character ?, either the letter Q (for "quoted printable") or the letter B (for BASE64), the character ? again, the encoded text, and finally the characters ?=. For example, the French word "voil<\#226>" is encoded =?ISO-8859-1?Q?V=F3il=E0?=. Incredibly inefficient, of course, but these will be used only a few times per message. Note that one extra restriction is put on quoted printable encoding, not present in the basic encoding: any question marks in the encoded text must be encoded. Otherwise, the sequence

<question mark><encoded octet> would be interpreted as the end of the encoded word.

Content Negotiation

As yet, there are no general standards, but HTTP 1.1 is an example of a protocol that provides facilities for the browser and server to negotiate the type of content to be provided. In particular, the browser can automatically specify the language and preferred encoding of content. The server may ignore this, if content in that language is unavailable. This method is certainly more convenient for users than providing links to translations in various languages.

Another example of content negotiation is provided by the MIME multipart/ alternative format. This format allows the same content to be presented in several ways. For example, a mail message can be formatted as both plain text and as HTML. Many UNIX mail user agents do not understand HTML, but Netscape certainly does. This allows “dumb” MUAs (or people who hate HTML e-mail) with a minimal understanding of MIME to read the e-mail as plain text, while those using Netscape to read their mail get the (dubious, in my opinion) benefit of the HTML presentation.

Conclusion

These two articles have presented an overview of the principles of internationalization. It hasn't been brief, but it is hardly complete or comprehensive. Linux is now in fairly good shape with respect to the basic facilities for internationalization with the wide dissemination of GNU libc version 2 (usually known on Linux systems as glibc or libc6).

A few issues still remain to be worked out, especially with respect to Asian languages. We can expect the standards to become more comprehensive over time. For example, locales may deal with line wrapping conventions, or the locale model may be extended to support multilingual applications directly.

However, the main effort today must be on the part of applications programmers and multilingual volunteers. Applications programmers need to use the POSIX locale facilities and GNU gettext to internationalize their programs. Multilingual volunteers should join the GNU translation project and help translate message catalogs for their favorite programs.

Resources

Stephen Turnbull (turnbull@sk.tsukuba.ac.jp) is an economist teaching and researching in Japan. He is excited about the way the Open-Source movement is turning conventional economics on its head, but is too busy playing with his Linux systems to do much economic analysis.

Archive Index Issue Table of Contents

Advanced search