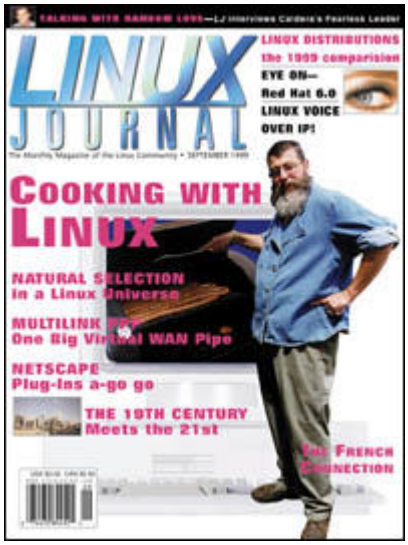
Linux Journal Issue #65/September 1999



Focus

Cooking with Linux *by Marjorie Richardson*

Features

Cooking with Linux —The French Connection *by Marcel Gagné*
Mr. Gagné provides us with several recipes from his famed French kitchen.

Natural Selection in a Linux Universe *by Travis Metcalfe and Ed Nather*
Astronomers at the University of Texas-Austin are using the ideas of Charles Darwin to learn about the interior of white dwarf stars—using a minimal parallel Linux cluster tailored specifically to their application.

Multilink PPP: One Big Virtual WAN Pipe *by George E. Conant*
MLPPP gives network managers the power to deliver WAN bandwidth on demand using an array of services.

Netscape Plug-Ins *by Larry Hoff*
Extending Netscape's ability to handle additional file formats.

Forum

Open Source with Applix *by Craig Knudsen*
Linux Distributions Comparison
Caldera's Ransom Love *by Marjorie Richardson*
Multicast: From Theory to Practice *by Juan-Mariano de Goyeneche*

Columns

Reviews

Departments

Strictly On-line

This article is about filtering, a very powerful facility available to every Linux user, but one which migrants from other operating systems may find new and unusual.

The Unified Modeling Language User Guide *by Geoff Glasson*

Archive Index

Advanced search

Focus: Cooking with Linux

Marjorie Richardson

Issue #65, September 1999

Being able to achieve our computing goals while having a good time is a winning combination for all of us.

Back in the early days of *Linux Journal*, we had a column by Matt Welsh called "Cooking with Linux" in which he told us about fun things to do with Linux, and when necessary, gave us scripts (recipes) for accomplishing them. With this month's focus, we revisit the spirit of those columns with articles from experts, who show fun things we can do with Linux. After all, being able to achieve our computing goals while having a good time is a winning combination for all of us.

So this month we feature articles about scripts (shell and Perl) for obtaining useful information from your computer, Netscape plug-ins, multilink PPP and clustering—all designed to help you have fun with Linux.

Other Things

Two things I'd like to mention, in case you haven't already found them while browsing the Web. One is *Linux Journal Interactive*, our archive site for all articles printed in our magazine from issue number one through the current issue. This site can currently be accessed only by *Linux Journal* subscribers; you will need your subscription number from the label to log in. In addition to the articles, you can post comments about them to discussion groups. Also available is a search facility to find *LJ* articles on any subject of interest to you. *LJI* can be found at http://interactive.linuxjournal.com/.

Two, we have completely redesigned our web site at www.linuxjournal.com/. The new site came alive on June 25. If you haven't visited us lately, now is the time to take another look. I know you will like the new appearance and the information you find there. This is the location for the Table of Contents of each issue with links to all "Strictly On-Line" articles. Unlike *LJI*, the articles here are

world readable. Accessible are product reviews, the Best of Technical Support column and at least one feature each month. Also on the site are articles by SSC staff members, including *Linux Journal* Sr. Editor, Doc Searls, as well as links to all the Linux resources you'll ever need.

Come on by, we'll be looking for you.

—Marjorie Richardson

Archive Index Issue Table of Contents

Advanced search

Cooking with Linux—The French Connection

Marcel Gagné

Issue #65, September 1999

Mr. Gagné provides us with several recipes from his famed French restaurant.

Allo, and welcome to *Chez Marcel*, home of fine French Linux cooking.

Please take a seat. If you have not already done so, I would like you to read this article with a somewhat exaggerated French accent since that is the way I wrote it. Even the README files are heavily accented. For those of you who, like myself, are French, you may translate as you read, thus adding another level of authenticity to the excellent menu that awaits you.

It is an honour and a privilege to welcome you to my kitchen. Tonight, I have a special treat for you. The recipes I have prepared are made from common Linux distribution ingredients so that you can recreate these delicacies when you return to your own kitchen CPU. Tonight's menu:

- Mail notification for Windows clients
- What's my IP?
- diald Control
- Waiter! More disk.
- Waiter! There's a fly in my system.
- Is it That Time Already?

A Little Wine with Dinner?

I must tell you right now that I am thrilled you have decided to join me in my kitchen. Much of what I have in store for you is designed to make Windows 9*x* more productive with a little help from Linux. If you have already cast away your Windows PC but you miss some forgotten piece of software (provided that software is not too demanding), you could check out http://www.winehq.com/ for the latest version of the Windows non-emulator for Linux.

For many users out there, the Windows workstation is still on the desktop, while the Linux machine performs such noble tasks for the Windows user as connecting to the Internet, providing firewall services, delivering mail, sharing disk space and printers, among other services.

What we have in store for you here today are Linux scripts that can make your Windows experience more enjoyable.

Talking to Windows

In the next few examples, we will have our Linux system talk back to Windows. While that is a good idea, we must first set up Windows to listen; otherwise, all our talk will disappear into the proverbial ether. When our Linux system needs to contact a Windows user, it will do so using Samba's **smbclient** messaging functionality. Obviously, that requires a system running Samba. On the Windows side, it requires Winpopup.

To set up Winpopup to start each time you boot Windows, create a shortcut in the Windows Startup folder. Click on the Start button, then Settings, then Taskbar. Choose the tab that says “Start Menu Programs”, then click the Advanced button. This will open a Windows Explorer window. Open the Startup folder under Programs and add a shortcut to winpopup.exe. Accept the default name, and you will wind up with a cute little Jack-in-the-box icon. Right-click on it to access the properties tab. Now, set the Run: option to *Minimized*. Winpopup starts up out of the way, and pops up only when it gets a message. Double-click the icon to start it right away.

One more thing: you should now see Winpopup sitting minimized on your taskbar. Click on it to bring it up. Now click on Messages, and choose Options from the drop-down menu. Click the box for “Pop up dialog on message receipt”. Click OK and re-minimize Winpopup. Windows is now ready to receive messages from your Linux system, which brings us to the next item on the menu.

Why Didn't Somebody Tell Me I Have Mail?

This is a frequent question we get from users. You are the administrator of a small- to medium-size office; you've set up an Internet gateway that picks up the mail on a regular basis using **fetchmail**, which then stores it on your Linux server. The problem is that your users are still running Windows with some sort of stand-alone POP3 e-mail package like David Harris' Pegasus Mail. While many packages can be set up to retrieve mail automatically, keeping them open and running taxes their already taxed Windows system. This means users tend to point out how slowly their systems are running. You suggest they close a few applications. “How about e-mail?” you suggest. To which they reply, “What if I

miss an important message?" The solution is the checkusermail script shown in Listing 1.

Listing 1.

In /usr/local/etc, create a small text file called mail_notify with the names of users who receive mail on your system. If the Windows clients are named differently (in the network configurator of the Windows control panel) than the user ID for mail, create an /etc/lmhosts file with IP addresses matching your mail user IDs, and the results should be the same. The script can be run from **cron** at whatever time interval suits your environment. Since it runs as root, it can spy into everyone's mailbox with the **frm** command. It will tell each user they have mail and how many messages. If there is no mail, there is no message and no need to waste time and energy checking every few minutes.

Another happy soul you'll discover after you deploy this script is the boss who had been complaining about either the amount of time users spent checking mail, or the user who did not check it often enough and missed an important message.

Hey There, What's Your IP?

In the last example, I talked about the beleaguered administrator (is there any other kind?) and suggested this might be a useful tool for them as well. What if you are administering that system from far away? Worse yet, this Internet gateway runs on a budget—it's a simple dial-up connection to an ISP running diald and IP masquerading. When trouble strikes, they call you. *Trouble* means you use TELNET to log in, check things out and fix the problems remotely. Trouble is, the IP is dynamic and changes with each connection to the ISP. The solution this time is the showppp.pl script shown in Listing 2. The script is simple and lives in cgi-bin. Depending on your web server setup, you might want to rename this one showppp.pl.

Listing 2.

Setting up a simple Intranet for your users is a breeze. A nice corporate home page with some popular links to the Internet makes it easier on your resources than having each user start up their browser with http://www.cnn.com/ or http://www.excite.com/ as their start page. If your dial-up connection gives you only a fixed number of hours per month, that time can get chewed up very quickly.

Why not use that same page and give your users a link to a script that will display the IP address of your dial-up IP connection without requiring them to log in to your Linux server? Figure 1 shows just such a page.

Figure 1. Show IP Page

Speaking of diald...

Every once in a while, the **diald** process may get hung up. Sometimes the modem has gotten hung up and diald can't seem to drop it. In either case, the easiest way to get life back to normal is to stop diald and restart the process. Since you are operating remotely and want to make this as painless as possible, you can either do the work yourself or talk one of your remote users through the process. It would be so much easier if they could just type one command, rather than doing a **ps ax | grep diald** while hoping they kill the right PID and not bring your system down. Listing 3 is a little script to do just that. The user simply types:

```
diald.control restart
```

to get things moving again. You can also use the script to stop diald, start it, or shut it down and start a single instance of mgetty in order to pick up a fax (but that's for another time).

Listing 3.

You can use this example as a template to create any number of start, stop or restart scripts. When necessary, you can have trusted users log in to your server and run simple administration scripts like this one without risking damage to your system or the equally ominous possibility of killing an important process like **init**. After all, **kill --1** looks very similar to **kill 1**, doesn't it?

Waiter! More disk!

Remember those great little messages for letting your users know they have mail? How about using that same technique to let you, the administrator, know that some important system event needs your attention? For example, suppose you are constantly battling disk space. Wouldn't it be nice if the system let you know that resources are low? With the next little Perl script (see Listing 4), you can have your Linux system send a pop-up message to your Windows workstation alerting you that system resources are low. This script scans disk partitions and reports to client **speedy** that disk space usage is over 90%. (See Figure 2.)

Listing 4.

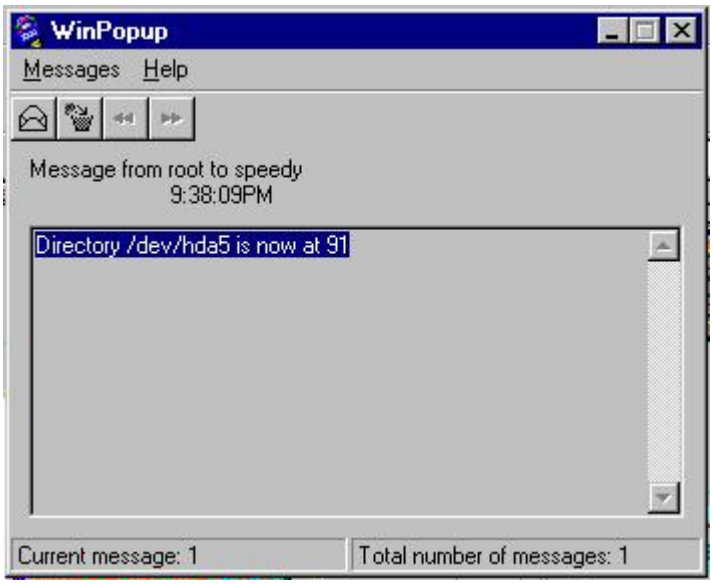


Figure 2. Disk Space Usage Screen

The Windows **winpopup** utility is a great tool for the user who still runs a Windows client but needs to get information sent to him from their Linux servers. This script and the earlier mail notification script highlight this flexibility.

By the way, the next item on the menu, the fly—that's just to get you thinking about other possibilities.

Where Does the Time Go and What Time Is It Anyway?

In my business, I've set up a number of Linux internet gateways. To make these installations as inexpensive as possible, we set the machines to dial every half hour or so to pick up mail. That half-hour mark is also our cue to log in using TELNET if we need to do some administration work on the machines. Unfortunately, the time on each machine can vary. How about a way to check the time against some reliable source and adjust the Linux server?

The script in Listing 5 is a bit of Perl magic. It opens a socket on a secondary time server, picks up the time, and closes the socket. This is a nice alternative to NTP in that it does not tax the NTP time server's resources. For that very reason, the NTP rules ask that you do not use primary servers if you don't need to. A link to the “Public NTP Time Servers” page is provided in Resources.

Listing 5.

Notice the line in Listing 5 that says **remote_host="chime.utoronto.ca";**. The host specified here is chime.utoronto.ca, but it could be one of any number of machines which offer primary or secondary time services. Consider the rules, though, and visit the NTP time server page for a system in your area and time zone.

Sorry Folks, it's Closing Time

Well, that last script on time was my way of getting all of you out of the restaurant. I hope you'll pay another visit to *Chez Marcel*. We hope we've succeeded in whetting your appetite for some of the wonderful and fun things you can do with a Linux system. Bon Appétit!

Resources

Marcel Gagné (mggagne@salmar.com) 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.

Archive Index Issue Table of Contents

Advanced search

Natural Selection in a Linux Universe

Travis Metcalfe

Ed Nather

Issue #65, September 1999

Astronomers at the University of Texas-Austin are using the ideas of Charles Darwin to learn about the interior of white dwarf stars—using a minimal parallel Linux cluster tailored specifically to their application.

Astronomers worry about how stars work. Our current models describe stars as huge, hot gasballs, bloated and made luminous by a fusion furnace deep inside that burns hydrogen into helium and releases energy in the process. A kind of internal thermostat keeps them stable, so our planet enjoys a comfortable environment in its orbit around our star, the sun. In about 6,000 million years or so, all available fuel will be burned up, and as the fuel gets low, the sun will bloat, then shrink until it is 100 times smaller than it is now, becoming a white dwarf star. Written inside, in the ashes of the furnace, will be its nuclear history.

We have pieced together this story by looking at many different stars, which last much longer than we do, but we cannot see inside any of them. Stars are very luminous yet thoroughly opaque. Geologists have built up a detailed picture of the earth's interior, even though it is opaque too; they do this by watching as compression waves from earthquakes rattle around inside and make their way back to the surface: seismology. By a very fortunate circumstance, we have found that some white dwarf stars vibrate internally with something akin to earthquakes, all the time. Their rapid changes in brightness tell us what is going on inside: asteroseismology.

To take advantage of this cosmic bonanza, we build computer models of the stars, with adjustable parameters that reflect, one-to-one, the physics going on inside. We must “vibrate” our model and tweak its parameters until the model behaves like a real star. We then believe that the parameters in our model tell

us about the physics inside the white dwarf star. We can then start to read the history written there.

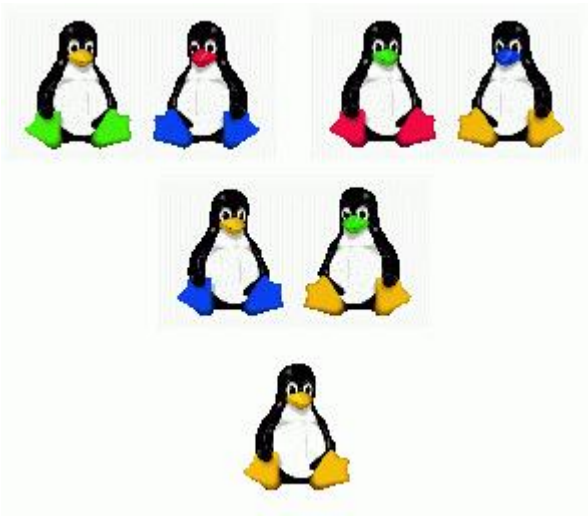


Figure 1. Evolving Penguin

Evolving Darwin

The basic idea is nifty, but the practice is a bit complicated. The models have many parameters, not all independent of one another, and we are not completely sure we have all the physics right. To make sure the set of model parameters we use is the best fit to the observed behavior and the only reasonable one, we have to explore a very large, multi-dimensional parameter space—far too large and complex to examine in exhaustive detail. No existing computer could handle it. There is a way though: we populate our huge parameter space at random with models whose parameters cover the whole shebang. Then we breed them together, preferentially allowing those which fit the observations fairly well to survive into later generations. This survival of the fittest is done with a "genetic algorithm" that mimics, in a crude but effective way, the process of natural selection proposed by Charles Darwin.

Genetic Algorithms

Even using this trickery, a *lot* of computing is required, so we built a massive parallel system to cut the runtime to hours instead of weeks. Most of the model calculations are done in floating-point arithmetic, so we measure performance in flops, the number of floating-point operations per second. Our assembled system, called a metacomputer, is capable of more than two gigaflops—2,000 million floating-point operations per second—not bad for an assembly of Linux boxes.

Our strategy in designing this system was minimalist; keep each computer node as cheap and simple as possible, consistent with doing our job and getting the maximum amount of computing for the buck. Our budget is fairly

limited. CPU cost is not a linear function of speed, so you pay a great deal more per megaflop for the fastest CPU on the market. Older CPUs are cheaper, but require more boxes and supporting electronics to house them for the same final performance. We watched the price drops with avid interest and jumped just after the 300MHz Intel P-II dropped below $300. We could afford a good master control computer and 32 computing nodes with our $22,000 budget.

Figure 2. Computer Lab

Some time after we settled on the design, we became aware of the existence of Beowulf machines through an article in *Linux Journal* (see Resources)—also parallel systems running Linux, but with faster Ethernet connections and more storage than our problem requires. They are much more general purpose than the system we built, so they can handle many problems ours cannot. They cost more too.

Cheap Hardware, Free Software

Our master computer is a Pentium-II 333 MHz system with 128MB SDRAM and two 8.4GB hard disks. It has two NE-2000 compatible network cards, each connected to 16 nodes using a simple 10base-2 coaxial network. We assembled the nodes from components obtained at a local discount computer outlet. Each has a Pentium-II 300 MHz processor housed in an ATX tower case with 32MB SDRAM and an NE-2000-compatible network card. We used inexpensive 32KB EPROMs, programmed with a BP Microsystems EP-1 using a ROM image from Gero Kuhlmann's Netboot package, allowing each node to boot from the network.

Table 1.

Configuring the software was not much more complicated than setting up a diskless Linux box (see Robert Nemkin's Diskless Linux Mini-HOWTO). The main difference was that we minimized network traffic by giving each node an

identical, independent file system rather than mounting a shared network file system. Since the nodes had no hard disks, we needed to create a self-contained file system that could be mounted in a modest fraction of the 32MB RAM.

To create this root file system, we used Tom Fawcett's YARD package (http://www.croftj.net/~fawcett/yard/). Although Yard was designed to make rescue disks, it was also well-suited for our needs. We included in the file system a trimmed-down, execute-only distribution of the PVM (parallel virtual machine) software developed at Oak Ridge National Laboratory (http://www.epm.ornl.gov/pvm/). PVM allows code to be run on the system in parallel by starting a daemon on each node and using a library of message-passing routines to coordinate the tasks from the master computer.

We configured the master computer to be a BOOTP/TFTP server, allowing each node to download the boot image—essentially a concatenation of a kernel image and a compressed root file system. We used the Netboot package (http://www.han.de/~gero/netboot/) to create this boot image using the root file system created by YARD and a small kernel image custom-compiled for the nodes.

How It Works

With the master computer up and running, we turned on each node one at a time. By default, the BIOS in each node tries to boot from the network first. It finds the boot ROM on the Ethernet card, and the ROM image broadcasts a BOOTP request over the network. When the server receives the request, it identifies the associated hardware address, assigns a corresponding IP address, and allows the requesting node to download the boot image. The node loads the kernel image into memory, creates an 8MB initial RAM disk, mounts the root file system, and executes an **rc** script which starts essential services and daemons.

Once all nodes are up, we log in to the server and start the PVM daemon. An rhosts file in the home directory on each of the nodes allows the server to start up the daemons. We can then run in parallel any executable file that uses the PVM library routines and is included in the root file system.

For our problem, the executable residing on the nodes involves building and vibrating a white dwarf model and comparing the resulting theoretical frequencies to those observed in a real white dwarf. A genetic algorithm running on the master computer is concerned with sending sets of model parameters to each node and modifying the parameter sets based on the results. We tested the performance of the finished metacomputer with the same genetic algorithm master program as our white dwarf project, but with a

less computationally intensive node program. The code ran 29.5 times faster using all 32 nodes than it did using a single node. Our tests also indicate that node programs with a higher computation to communication ratio yield an even better efficiency. We expect the white dwarf code to be approximately ten times more computationally intensive than our test problem.

Stumbling Blocks

After more than three months without incident, one of the nodes abruptly died. As it turned out, the power supply had gone bad, frying the motherboard and the CPU fan in the process. The processor overheated, shut itself off, and triggered an alarm. We now keep a few spare CPU fans and power supplies on hand. This is the only real problem we have had with the system, and it was easily diagnosed and fixed.

Conclusions

The availability of open-source software like Linux, PVM, Netboot and YARD made this project possible. We would never have considered doing it this way if we'd had to use a substantial fraction of our limited budget to buy software as well as hardware and if we'd been unable to modify it to suit our needs once we had it. This is an aspect of the Open Source movement we have not seen discussed before—the ability to try something new and show it can work, before investing a lot of money in the fond hope that everything will turn out fine.

Resources

Travis Metcalfe (travis@astro.as.utexas.edu) is a doctoral student in astronomy at the University of Texas-Austin. When not sitting in front of a computer, he can usually be found tilting at windmills. His use of Linux since 1994 has reportedly made him more unruly.

Ed Nather (nather@astro.as.utexas.edu) is a professor of astronomy who publishes science fiction in several astronomical journals, under an alias (R. E.

Nather). In his spare time, he installs and re-installs the newest Linux distributions, hoping to find the perfect one. He also believes in the tooth fairy.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Multilink PPP: One Big Virtual WAN Pipe

George E. author)

Issue #65, September 1999

MLPPP gives network managers the power to deliver WAN bandwidth on demand using an array of services.

Network management is a little like alchemy: take a dash or two of ISDN, add some frame relay, throw in a couple of routers, mix them all together, and somehow, some way, the result is bandwidth gold.

Of course, the formula for creating fully interoperable networks is much more complicated. Fortunately, network managers do have access to some tools that can make bandwidth magic a little easier to perform. Two of the most important elements in the technology bag of tricks are the point-to-point protocol (PPP) and its follow-up, the multilink point-to-point protocol (MLPPP).

PPP, a product of the Internet Engineering Task Force (IETF), is the de facto WAN link protocol for connecting clients and servers and for interconnecting routers to form enterprise networks. PPP's main advantage is that unlike other protocols which operate at the data link layer, PPP achieves interoperability between devices by negotiating different configuration options, including link quality, link authentication and network protocols.

Over the years, the IETF has made some significant changes to PPP. But as its name states, PPP is intended for simple point-to-point connections. Now that the enterprise network infrastructure is moving rapidly to digital switched services such as ISDN, frame relay and ATM, PPP is in need of even more changes.

Enter MLPPP, known in IETF circles as RFC (Request for Comment) 1717. MLPPP takes advantage of the ability of switched WAN services to open multiple virtual connections between devices to give users extra bandwidth as needed. With MLPPP, routers and other access devices can combine multiple PPP links connected to various WAN services into one logical data pipe.

The IETF formally approved the MLPPP specification last November. Makers of ISDN routers and access devices have already started using MLPPP to bundle 64Kbps ISDN B channels to deliver more bandwidth. MLPPP also allows network managers aggregate WAN circuits of different types without requiring major configuration changes to existing router Internet works.

Because MLPPP works over any switched WAN service, it has a wide range of potential uses (see “PPP Plus”). Network managers could deploy MLPPP-equipped devices to create a technology-independent enterprise framework in which the actual WAN services linking two devices would be invisible to end users. Under this model, WAN devices would negotiate bandwidth rules between two directly connected peers, using whatever type of service was available. New digital WAN services such as ATM (asynchronous transfer mode) could be added to the network mix as needed, without making the existing network infrastructure obsolete.

The ABCs of PPP

Although it is usually considered a single entity, PPP is actually a group of protocols that together provide an extensive set of network connectivity services. The PPP suite is based on four key design principles: negotiation of configuration options, multi-protocol support, protocol extendibility and WAN service independence.

Negotiation of configuration options: This refers to PPP's ability to establish throughput requirements between two directly connected end systems. In an enterprise network, end systems often differ in terms of buffer requirements, packet-size limits and network protocol-support lists. The physical link that interconnects any two end systems could vary from a low-speed analog line to a high-speed digital connection with varying degrees of line quality.

To cope with all these possibilities, PPP has a suite of standard default settings to handle all common configurations. To establish a link, two communicating devices attempt to use these default settings to find a common ground. Each end of the PPP link describes its capabilities and requirements; the settings are negotiated between the two sides for each option at the link level. These options include data encapsulation formats, packet sizes, link quality and authentication.

The protocol that negotiates all these options is known as the link control protocol (LCP). The protocol that negotiates the network protocols to be multiplexed over a PPP link is called the network control protocol (NCP); there can be many NCP data streams over a single PPP link. Although PPP's configuration negotiation options also allow end systems to set link peer authentication (a security function) and data compression options, PPP does

not dictate the actual algorithms used for security or compression. For security, PPP defines PAP (password authentication protocol) and CHAP (challenge handshake authentication protocol) as common standard authentication methods that may be negotiated, but it also lets users add new authentication algorithms. The same holds true for compression.

Multi-protocol support: PPP's ability to handle multiple network-layer protocols was one of the chief reasons it became a de facto standard. Unlike the serial IP protocol (SLIP), the IETF routing protocol that handles only IP datagrams, PPP works with a range of packet formats including IP, Novell IPX, AppleTalk, DECnet, XNS, Banyan Vines and OSI. Each network-layer protocol is separately configured by the appropriate NCP.

Protocol extendibility: Over the years, the IETF extended PPP through a number of additional RFCs that define features like common data authentication services and encryption capabilities for security and compression algorithms. For example, with many WAN technologies, compression algorithms are chosen according to the quality of the link. Different technologies use different compression schemes, introducing multiple layers of compression and decompression into the network. Running PPP compression at the NCP level removes these considerations and uses fewer system resources.

WAN service independence: The initial version of PPP was built expressly to run over HDLC (high-level data link control) networks. Since then, the IETF has added RFCs that enable PPP to work with every major WAN service now in use including ISDN, frame relay, X.25, Sonet and synchronous/asynchronous HDLC framing.

The Need for MLPPP

For all its strengths, PPP has one inherent limitation when it comes to network deployment: it is designed to handle only one physical link at a time. MLPPP does away with this restriction. MLPPP is a higher-level data link protocol that sits between PPP and the network protocol layer. It accommodates one or more PPP links, with each PPP link representing either a separate physical WAN connection or a channel in a multichannel switched service, such as ISDN or frame relay.

MLPPP's ability to combine multiple lower-speed links into a single, higher-speed data path is often referred to as WAN-independent or packet-based inverse multiplexing (see “WAN Independence” below). Packet-based inverse muxing isn't new; for instance, ISDN vendors have been offering ways to combine multiple ISDN 64Kbps B channels for some time. But up to now, these solutions have been proprietary: vendor and technology-specific. MLPPP

embodies a standard approach that cuts across vendor and WAN technology lines.

MLPPP negotiates configuration options the same way as conventional PPP. However, during the negotiation process, one router or access device indicates to the other communicating device that it is willing to combine multiple connections and treat them as a single physical pipe. It does this by sending along a multilink option message as part of its initial LCP option negotiation.

Once a multilink session is successfully opened, MLPPP at the sending side receives network protocol data units (PDUs) from higher-layer protocols or applications. It then fragments those PDUs into smaller packets, adds an MLPPP header to each fragment and sends them over the available PPP links (see Figure 1). On the receiving end, the MLPPP software takes the fragmented packets from the different links, puts them in their correct order based on their MLPPP headers and reconverts them to their original network-layer PDUs.

Figure 1. The Multiprotocol Link

MLPPP is independent of the actual physical links and the WAN services that run over them. This means MLPPP traffic can traverse a mix of physical and logical connections from multiple WAN services—a frame relay virtual circuit, multiple ISDN channels and an X.25 connection, for example. MLPPP functions as a logical link layer that dynamically adds or removes links between two communicating devices as bandwidth needs change. The MLPPP standard does not dictate how traffic is balanced over these member PPP links, leaving network managers free to determine how to use the available links or channels.

MLPPP's ability to combine separate PPP links into one logical data pipe is one of the most important features of the protocol. It allows additional WAN bandwidth or new WAN services to be added as needed without disrupting the existing WAN infrastructure. With MLPPP, different WAN services such as ISDN, frame relay and ATM can be used together. For instance, a network manager can establish a frame relay connection to serve as the primary link between a central site and a branch office, with ISDN serving as an adjunct when bandwidth demand rises (see Figure 2).

Figure 2. Many Circuits, One Pipe

Through the dynamic addition and deletion of PPP links, MLPPP enables dynamic bandwidth allocation, or “rubber bandwidth”, between two peer systems. During the LCP option negotiation, all PPP links in an MLPPP group identify themselves as belonging to the same group or bundle. To add a new link or WAN service to the bundle, all that's required is attaching the

appropriate MLPPP group identifier to the link. Likewise, when a member PPP link is terminated, it is automatically removed from its parent MLPPP bundle by eliminating the identifier.

PPP is WAN service-independent, so the member links of an MLPPP bundle can be associated with either permanent virtual circuits (PVCs), which have fixed end points, or switched virtual circuits (SVCs), which are dialed up on demand.

MLPPP's ability to create different groups of WAN links produces some intriguing possibilities for network managers. For instance, they could use MLPPP to segregate traffic according to the network-layer protocol. This approach would enable network managers to separate expedited control messages from normal data traffic or to queue data into separate MLPPP bundles based on application-specific requirements.

Here's an example of how MLPPP's segregated packet queueing works. Suppose a central site is connected to a remote site via two 64Kbps frame-relay links and two ISDN basic-rate interface (BRI) connections. Two types of traffic traverse these links: IP traffic from UNIX operations and DECnet traffic from a Digital Equipment Corporation VAX network. If the frame relay and ISDN channels are treated as one MLPPP bundle, both traffic types have access to the full bandwidth of the link at any given time.

The single-pipe approach makes for easier network management, but it could create problems if one traffic type starts dominating the pipe. In this example, if the UNIX IP traffic started bursting beyond 60 percent of the overall link rate, it would begin to eat into bandwidth available for DECnet, slowing performance for users on the VAX network.

With MLPPP, this problem can be avoided. The network manager can not only combine various physical interfaces to create one large pipe, but also allocate channels within that virtual pipe. For instance, the network manager can create two 128-kbit/s MLPPP bundles, each consisting of a single ISDN B channel and a 64Kbps frame-relay link. Those bundles could then be dedicated to each type of traffic.

Spoofing

One big problem with using routers for switched WAN services isn't activating a link, but shutting it down when data transfer is done. Most LAN protocols and client-server applications are chatty, carrying on almost incessant messaging to synchronize routing databases and maintain client-server sessions.

Left unchecked, these processes can keep an ISDN link up indefinitely without passing a single byte of application data. Needless to say, all this uptime quickly

adds up, especially where charges are based on call duration. Considering that more than 35 percent of WAN costs are related to line costs, the ability to control activation and deactivation of member links in an MLPPP group is crucial.

MLPPP offers two solutions: usage thresholds and spoofing. In the usage threshold scheme, once a circuit becomes idle or the traffic it carries falls below a level predefined by the network manager, MLPPP will automatically remove that circuit from its bundle until demand rises. The problem with the usage threshold approach is that it can be difficult to define threshold levels effectively in bursty environments using chatty protocols. For example, in Novell IPX environments, it can be difficult to gauge the requirements of SAP (service advertising protocol) and RIP (router information protocol) messages.

Spoofing helps address this problem. It is a technique used by routers to filter network traffic. Spoofing keeps unnecessary traffic like session keep-alive messages from traversing the WAN link. Rather than sending these messages over the WAN, the router acts as a proxy and responds to them locally. Once the router takes over the polling and responses for the application, the WAN link can be shut down until it is truly needed.

WAN Magic

MLPPP's WAN service independence means users and network managers can be insulated from network service changes. As new WAN services, such as frame relay and ATM become available, MLPPP can be used to incorporate them into logical bundles. To routers, MLPPP looks like a data link protocol; the router doesn't have to deal with the complexity of the various physical connections and switched circuits that MLPPP draws together in its logical pipes. This helps reduce router reconfiguration costs, since a new router interface isn't required when a new WAN service is added.

To network managers, the difference between adding a new circuit or virtual circuit to an MLPPP bundle and adding a router interface is significant. Adding a circuit to a preexisting MLPPP logical pipe is transparent to the network, particularly in switched environments like ISDN, frame relay or ATM. It simply adds bandwidth to the pipe; no additional interfaces or path information is required. In contrast, any change to a physical router interface triggers an update to the routing table of every router involved in the change. In environments where circuits are frequently activated and deactivated, this could generate excessive amounts of network topology changes and much extra work for network managers.

Not only can MLPPP save network managers time and effort, but it also offers an important tactical tool for network designers. It can be used to simplify fault management and build redundancy into the network at low cost.

Keeping Up with ATM

Along with making use of WAN services already in place, MLPPP is positioned to work with technologies just making it to the real world. The most prominent of these technologies is ATM.

ATM SVCs can be activated and deactivated on demand, much like ISDN circuits. ATM circuits can be included in an MLPPP circuit group as more bandwidth is required. Bundling will become especially useful if lower-speed ATM ports (T1 or T3) become widely deployed.

In the long run, as ATM takes over the enterprise network backbone, things could get more complicated. ISDN, frame relay and ATM will dominate the WAN landscape, with ISDN and frame relay functioning as a feeder technology and ATM serving as the enterprise backbone aggregating ISDN and frame-relay circuits over faster pipes. An ATM pipe at Sonet OC3 speed (155Mbps) can aggregate more than 2,400 64Kbps ISDN B channels.

That is a large amount of bandwidth by today's standards, but thanks to the rise of LAN switching and high-speed LANs, aggregate throughput requirements for the LAN will escalate at an even more rapid rate, reaching tens of gigabits per second in the next few years. To reduce the disparity between the LAN and WAN worlds, network managers will need to aggregate B channels and frame-relay circuits. Inverse multiplexing using MLPPP offers a flexible way to do this.

PPP Plus

MLPPP delivers some key functions to help network managers build multi-protocol enterprise networks. Here are some of MLPPP's strongest selling points:

Mix-and-match WANs: With MLPPP, net managers can configure multiple physical links and virtual circuits as one logical pipe, using different WAN service types (ISDN, frame relay, X.25 and ATM) in the process.

Bandwidth on demand: Circuits can be activated and added automatically to a logical pipe when more bandwidth is needed, or if one of the links in the logical pipe fails.

Big protocol tent: MLPPP handles routing for all major network- and transport-layer protocols, including IP, IPX, Netbios, DECnet and SNA.

Network negotiation: MLPPP has inherited PPP's ability to negotiate configuration options between communicating devices. This means two end systems can set the terms of transmission without requiring manual intervention.

No traffic, no link: MLPPP uses LAN protocol spoofing to keep unnecessary network traffic, such as session keep-alive packets, from traversing the WAN.

WAN Independence

The two basic types of inverse multiplexing are circuit-based and packet-based. Under the circuit-based scheme, a data stream is sliced into equal portions, regardless of its contents, with each portion transmitted over a different available circuit. With circuit-based inverse muxing, synchronization of traffic streams is generally handled by hardware.

Packet-based inverse multiplexing is a software-based process. In this scheme, packets are distributed among available circuits according to rules or policies that govern allocation of traffic across circuits. These rules can include segregation according to protocol or percentage-based prioritization. The multilink point-to-point protocol (MLPPP) uses packet-based inverse multiplexing.

One key difference between the two approaches is that packet-based inverse multiplexing schemes can handle multiple circuit types, while circuit-based schemes cannot. With MLPPP, for example, a synchronous 56Kbps X.25 link can be bundled together with a 64Kbps ISDN B channel to create a single logical pipe offering 120Kbps of bandwidth. This bundling ability extends to the full range of available WAN services, including dial-up analog lines, switched 56Kbps services, frame-relay connections and T1 or T3 services.

Another difference is that packet-based inverse multiplexing is seamless to the destination router or end system. In the case of MLPPP, the flow of data appears consistent by means of software synchronization of MLPPP fragments. In contrast, circuit-based inverse muxing requires the receiving device to stop data flow until all bonded channels are characterized for latency. This is the only means by which the hardware-based solution can ensure data is received in the correct order.

Except for applications that require constant bit rate transmission or have strict requirements on circuit latency, software-based solutions are generally regarded as superior because they add considerable value and flexibility to the

basic inverse muxing function. In the case of MLPPP, users are able to combine different WAN connections while taking advantage of PPP's configuration negotiation and multi-protocol routing support.

This article was originally published by Data Communications and can be found on the web at http://www.data.com/tutorials/multilink_ppp.html.

George E. Conant is a cofounder of Xyplex Inc. (Littleton, Mass.), a maker of internetworking products.

Archive Index Issue Table of Contents

Advanced search

Netscape Plug-Ins

Larry Hoff

Issue #65, September 1999

Extending Netscape's ability to handle additional file formats.

Plug-ins are a powerful mechanism for extending the capabilities of Netscape's web browser. Using plug-ins, the browser can display files in formats that were not even conceived of when the browser was developed, such as multimedia files embedded within larger web pages. This allows web pages to be designed with maximum visual impact. Like any powerful technology, plug-ins can easily be misunderstood or misused. This article will explore when Netscape plug-ins might be appropriate to use, explain how to install and remove plug-ins, suggest where to locate useful plug-ins, and provide insight into implementing your own plug-ins.

Plug-ins vs. Helper applications

It might be useful to compare and contrast plug-ins with helper applications. Plug-ins and helper applications share many features and may be used interchangeably in many circumstances. Helper applications are older and may be more familiar to users than plug-ins. Netscape has supported helper applications since version 1.0 of their browser software, but plug-in support did not show up until version 2.0 (version 3.0 for UNIX).

Both helper applications and plug-ins are supplementary software used by web browsers to handle specific file formats. Helper applications can be run independently of the web browser. Plug-ins, however, are integrated into the browser and can be run only within a browser. Potential helper applications may already be installed on your system. Some very useful helper applications, such as **xv**, were developed even before web browsers became popular.

Helper applications, being independent of the browser, must create their own window for a user interface. Plug-ins may use the window provided by the browser. Since they share the browser window, they can be used to display files

that are embedded within larger HTML files. Helper applications cannot display embedded files. Plug-ins can have access to file contents while the file is being downloaded by the browser. Such plug-ins are called “streaming” plug-ins. Helper applications are “launched” only after the file is fully downloaded.

Review of MIME

The MIME (multi-purpose Internet mail extensions) protocol, described in RFC-1521, allows applications to exchange different types of files on the Internet. A text header identifies the data format of the message body. Web browsers use MIME information to categorize files before displaying them. For example, if the MIME header indicates the body is of type “image/gif”, the browser will display the message body as a GIF (Graphics Interchange Format) file. If no MIME header is available, the browser assigns a MIME type to the file based on the file's extension. Once the MIME type of the file has been determined, the browser searches its internal tables for the plug-in or helper application assigned to that MIME type.

When to Use Plug-ins

As a web surfer, you may encounter HTML pages that have other files embedded within them. Without an appropriate plug-in, your browser will not be able to display the embedded file. It might seem impolite for a web-page designer to create a page you cannot display without specialized software, but there are some reasonable excuses for such behavior. For example, certain non-proprietary MIME types are in common use on the Internet, and are supported by Netscape on the Mac OS and Windows, but not UNIX. UNIX users must install a plug-in for such MIME types. One example of this type of file is MIDI (musical instrument digital interface); these are often embedded within web pages to provide background music.

As a web designer, you may want to use a newly developed multimedia file format on your web page. The current MIME type may have been developed too recently to be supported by existing browsers. The MIME protocol is designed to be extensible. Third parties are constantly developing new MIME types with specialized functionality. Examples of such MIME types include document rendering (application/pdf), portable graphics (image/png), vector graphics (application/shockwave-flash) or streaming audio (audio/pn-realaudio-plugin). Often these MIME types are developed in the hope of selling authoring software. Polite third parties provide free plug-ins for as many platforms as possible. Unfortunately, UNIX support is often the first to be sacrificed.

It may be tempting to invent new MIME types, rather than using existing functionally equivalent MIME types, or to embed specialized file types on your web page, such as Microsoft PowerPoint files. However, your viewers might not

have an appropriate plug-in or may be unable or unwilling to locate and install one. Your carefully designed web page may have dull gray rectangles where you expect flashy graphics. It is generally a good idea to use only the most popular Internet MIME types on your web pages. If your needs can be met only by using a more specialized MIME type, it is polite to first check for plug-in availability or even supply the necessary plug-ins yourself.

Where to Get Plug-ins

Many sources are available for Netscape plug-ins. Netscape maintains a web site of plug-ins organized by functionality or platform (see Resources 1 and 2). Netscape equips their browsers with a special plug-in called the “Default Plugin”, which is invoked when an unregistered MIME type is encountered. With user approval, the default plug-in will search the Netscape web site for appropriate plug-ins.

The Netscape web site is not the only source of plug-ins, although it may be the most convenient. Traditional sources such as Usenet newsgroups, WWW searches and even word of mouth can turn up useful plug-ins.

Be aware, however, that the same precautions should be used when downloading plug-ins as when downloading any other software. Although plug-ins are not full-fledged application programs, they can inflict just as much damage, intentionally or not. Unlike helper applications, plug-ins can cause the browser to leak memory, become unresponsive or even dump core. If the source code is available, you could inspect the code before compiling the plug-in yourself. Otherwise, you will have to trust the source of the plug-ins. Just because a plug-in is registered with Netscape's web page does not mean that Netscape provides any sort of warranty about the behavior of the plug-in.

How to Install and Remove Plug-ins

Plug-ins are dynamic code modules, native to the platform on which the Netscape client runs. For example, Windows plug-ins are DLLs and UNIX plug-ins are shared object libraries. When Netscape Navigator starts up, it checks for plug-in modules in certain directory trees. Each plug-in candidate in the directory tree is loaded, its capabilities determined using the plug-in API (application program interface), then unloaded. Internal tables assign MIME types to particular plug-ins. Later, if the browser encounters a MIME type that requires plug-in support, the plug-in is reloaded and remains loaded until the page is closed. The list of registered plug-ins can be viewed using the “Help/ About Plug-ins” menu. Removing a plug-in is as simple as deleting the shared library.

The README file which comes with Netscape Communicator for UNIX explains the algorithm for generating the plug-in list:

```
if($NPX_PLUGIN_PATH environment variable is set)<\n>
   Look at $NPX_PLUGIN_PATH, where
   $NPX_PLUGIN_PATH is a colon-delimited
   list of directories.
   else
   Look at all the following directories in
   order, overriding previous entries in case of
   duplicates:
      /usr/local/lib/netscape/plugins
      $MOZILLA_HOME/plugins
      $HOME/.netscape/plugins
```

The algorithm for Netscape 3.0 is even simpler. Only the directories /usr/local/lib/netscape/plugins and $HOME/.netscape/plugins are checked.

Only one plug-in or helper application can be assigned to each MIME type. These assignments are stored in the files $HOME/.mime.types and $HOME/.mailcap where they can be used by other applications. A dialog window allows users to resolve conflicts. For Netscape 3.0, the dialog is under the “Helpers” tab of the “Options/General Preferences” pull-down menu. For Netscape 4.0, the dialog is available via the “Edit/Preferences/Navigator/Applications” menu option. This menu also allows the user to associate file extensions with MIME types. Normally, the plug-in associates file extensions with MIME types when the plug-in is assigned to the MIME type.

Creating Plug-ins

You might have an idea for a great new Internet file format, but are wondering how to get Netscape to recognize it. You could be a web surfer whose favorite web site uses a file format unrecognized by Netscape or any of the available plug-ins. Or, you may just want to understand more about how your web browser works. In any case, the next few sections will briefly describe the process of designing and implementing your own plug-in.

First, a reminder about helper applications. If the file is not embedded within an HTML file, a helper application could also be used to display it. Helper applications are developed using traditional means, do not need to adhere to any special API, and do not require special debugging techniques. Even if your ultimate goal is a plug-in, it might be more efficient to first implement a helper application, then convert the helper application to a plug-in.

Once you have decided to build a plug-in, you will want to download Netscape's plug-in SDK (software development kit) (see Resources 3). The plug-in SDK includes documentation, example code and even a template plug-in, written in C, complete with Makefile. The SDK documentation includes a complete reference manual for the plug-in API, and some general guidelines for plug-in

design. Rather than duplicate that information here, I will explore how to use the API most effectively.

The Netscape SDK is designed to facilitate cross-platform development. The SDK allows developers to use a single source tree for UNIX, Windows and Mac OS plug-ins. However, there are significant hurdles for the cross-platform plug-in developer. Different GUI standards, OS standards and device interfaces must be considered. Even the plug-in file format varies by platform. For example,Windows plug-ins must have names beginning with “np” and provide descriptive information via a version resource, rather than the API. I will skirt around the thorny issue of cross-platform development by focusing on UNIX-only plug-in development.

The API

The plug-in API consists of two sets of functions. The first set, where the names begin with “NPP_”, are functions that the plug-in must implement. These functions will be called by the browser as it downloads the file. The second set, with names beginning with “NPN_”, are services which the plug-in may ask the browser to provide, such as allocating and freeing memory, reading and writing URLs, providing version information and writing messages to the browser status field.

There are more than a dozen “NPP_” functions. The thought of implementing such a large set of complex functions may seem scary. To demystify the API, these functions can be broken down into a few general categories. There are functions that allow the plug-in to describe its capabilities (NPP_GetMIMEDescription, NPP_GetValue), initialize and finalize data structures (NPP_Initialize, NPP_Shutdown, NPP_New, NPP_Destroy), write into a graphics area (NPP_SetWindow) and receive data (NPP_NewStream, NPP_Write, NPP_WriteReady, NPP_StreamAsFile, NPP_DestroyStream). There are also functions that allow the plug-in to enable LiveConnect (NPP_GetJavaClass) and to describe its graphics area to a printer (NPP_Print).

UNIX plug-ins must implement NPP_GetMIMEDescription. This function returns a semicolonseparated-list of MIME descriptions. Each MIME description includes the MIME type, the file extensions associated with that MIME type, and a brief description of the MIME type. NPP_GetValue returns the name of the plug-in, as well as a detailed description of the plug-in.

NPP_Initialize and NPP_Shutdown are called just after the plug-in is loaded and just before the plug-in is unloaded, respectively. These functions give the plug-in the opportunity to allocate and initialize global data structures, then free any allocated resources when they are no longer needed.

NPP_New and NPP_Destroy are called to create or destroy a particular instance of the plug-in player. More than one embedded file may have the same MIME type on a single HTML page. The plug-in may need to maintain separate data structures for each instance. When a plug-in instance is created, the browser provides environment information, such as whether the plug-in is embedded or full page, and whether there are any special directives within the HTML <EMBED> tag. However, the browser will not provide a graphics area or the file contents to the plug-in instance until after the instance has been successfully created.

NPP_SetWindow provides the plug-in with a graphics area to draw in. UNIX plug-ins are provided with a Motif Drawing Area widget. The plug-in may draw directly into the graphics area using X Window System functions, or it may create new widgets, using the Drawing Area widget as the parent. Note that because of the two-phase widget deletion scheme of X, the plug-in must not be linked with any widget library. Otherwise, X may try to execute the second deletion phase after the plug-in (and the widget library) has been unloaded, resulting in a core dump. The plug-in must use only widget classes already linked into the browser. For Netscape, this means Motif widgets.

The browser uses the functions NPP_NewStream, NPP_Write, NPP_WriteReady, NPP_StreamAsFile, and NPP_DestroyStream to negotiate with the plug-in about data transfer mechanisms. The plug-in can elect to receive the data piecemeal, or the plug-in may ask the browser to collect all the data into a file before delivering it. If the data arrives piecemeal, the plug-in can set upper limits on size and rate for data transfer from the browser.

The function NPP_GetJavaClass allows the plug-in to enable LiveConnect. LiveConnect is a technique for plug-ins to interface with Java and JavaScript. LiveConnect allows JavaScript to control the execution of a plug-in. LiveConnect requires a special Java class to be loaded and executed. For many plug-ins, this may be unnecessary overhead.

The function NPP_Print allows the plug-in to describe itself, in a platform-specific manner, to a printer. For UNIX, this means writing PostScript to a file. Full-page plug-in instances are given a choice of whether they would like to handle all aspects of printing or whether the browser should handle most aspects. Embedded plug-in instances have no choice. The browser will query the user about print destination, page size and orientation, etc. The browser will then open a file and begin writing its own PostScript. When the plug-in instance is encountered, the function NPP_Print is called. The browser passes in parameters reminding the plug-in instance of its location on the page and providing the plug-in instance with a FILE pointer to add its PostScript. When

NPP_Print returns, the browser continues to add PostScript to the file, then closes the file and sends it to the print destination.

If the plug-in does not add any PostScript, there will be a blank area in the plug-in's location on the printed page. Plug-ins that display still graphics should probably implement this NPP_Print. Plug-ins that display animation or which play sound may choose not to implement NPP_Print. This function will be called in response to a user selecting either the “File/Print” or the “File/Save As” menu option, then selecting PostScript as the output format. Needless to say, this is not a trivial function to implement properly. Even the PDF plug-in from Adobe contains a flaw which causes a core dump when using the “File/Save As” menu option. It may be better to omit this feature, rather than implement it poorly.

Design Issues

The SDK documentation advises against “blocking” any API call. The browser will not be responsive to user input while the API calls are executing. Therefore, these functions should complete in a reasonably short period of time. If a plug-in needs to perform time-consuming processing, a number of techniques can be employed. The best technique to use depends on a number of factors, including the nature of the plug-in as well as personal preference.

One popular technique is for the plug-in to create a completely separate “companion process”. This solution can be quite robust. A catastrophic failure in the companion process will likely not affect the browser. However, elaborate interprocess communication mechanisms may be needed to provide synchronization between the browser and the companion process. This technique is most appropriate if the processing is only loosely connected to the browser. It is also often the easiest technique for converting a helper application into a plug-in.

Alternatively, the plug-in could use time-slices within the browser to accomplish its processing. The plug-in can use the NPP_WriteReady function to limit the data transfer rate from the browser. With the data rate limited, the plug-in can perform the processing within the NPP_Write function without degrading browser performance. For example, a streaming video plug-in might limit the data transfer rate to the frame rate. Each time the browser invokes the NPP_Write function, the plug-in would need to process only one frame's worth of data before returning control to the browser. This technique is most appropriate for streaming plug-ins.

The X event processing loop can also provide processing time-slices. The X event processing loop acts much like the scheduler in a cooperative multitasking operating system. It can be commanded to perform small processing tasks during idle times, or after specific time intervals via the

XtAppAddWorkProc and **XtAppAddTimeOut** functions, respectively. This technique is most appropriate for plug-ins with interactive graphics, especially animation.

Finally, the plug-in could create a separate, asynchronous thread within the process. Thread programming can be difficult. Many UNIX libraries, especially graphics libraries, are not thread-safe. The plug-in designer must use care to avoid reentrancy problems. This technique should be reserved for situations where none of the previous techniques can be used.

Plug-ins can be difficult to debug. The browser does not load the plug-in until just before executing the plug-in functions. This does not leave much opportunity for a debugger to set breakpoints. It may be necessary to resort to using **printf**. One technique for using a debugger is to insert an artificial delay in a convenient location, such as at the beginning of NPP_Initialize. The delay can give a debugger time to “attach” to the browser. Once the debugger is attached, breakpoints can be set within the plug-in.

Conclusion

Netscape plug-ins can enhance the web surfing experience. After all, it is much more fun to experience creative multimedia than it is to see dull gray rectangles. Linux plug-ins are available for many commonly used MIME types; some require compiling, others are available as shared libraries, simplifying installation. Implementing plug-ins for unsupported MIME types is well within the capabilities of an experienced Linux programmer and can be fun. The source code for the UNIX MIDI plug-in (UMP) is available on the UMP download page (see Resources 4). This source code can be used as a starting point for other plug-in projects. I glossed over cross-platform development, LiveConnect support and printing issues. For more information on these topics or any plug-in topic, feel free to contact me.

Resources

Larry Hoff works for Brookhaven National Lab, where he develops embedded software for particle accelerator control. He can be reached at hoff@bnl.gov.

Archive Index Issue Table of Contents

Advanced search

Open Source from Applix

Craig Knudsen

Issue #65, September 1999

A look at Applix's open-source initiative—what they are doing and why.

Applix, Inc. is best known for Applixware Office, a cross-platform suite of desktop productivity tools. (See review by Dean Staff in this issue.) Founded in 1983, Applix's original goal was to develop and market multi-user graphical UNIX-based integrated office applications. Since then, Applix has become very popular among those in the engineering, government and financial sectors. Applixware became available for Linux in September of 1998. In March 1999, Applix announced its first Open Source initiative with SHELF.

The Extension Language Facility (ELF) enables developers to integrate applications and corporate data with Applixware Office. For example, all user interfaces for the Applixware applications (Words, Graphics, Spreadsheets, Presents, Mail and Data) are all built with ELF. SHELF (SHared ELF) is the open-source release of the ELF developer tools including Applix Builder, an object-oriented, graphical IDE (Integrated Development Environment) released under the GNU Library General Public License (LGPL). (See Figures 1 and 2.)

Figure 1. Applix Builder Screen

SHELF can be used to rebuild the ELF shared library which is used by newer versions of Applixware. This allows you to extend ELF's capabilities and take advantage of those changes from within Applixware.

Like Java, ELF is a platform-independent language that does not need to be recompiled on each platform. In general, ELF applications execute slower than C or Java. Computation-intensive functions are best implemented as C add-ins to ELF. ELF's loose type checking is an advantage in smaller applications, but can become a disadvantage for large-scale applications. Unlike C, ELF is fully memory-safe and includes error signalling. Developers are free to focus on

algorithms and the user interface rather than memory management. ELF also provides simple and robust interfaces to relational databases.

Applix is hoping ELF will become a popular language for layered applications and lightweight application-building where development cycles are measured in person-days to person-months. They do not anticipate ELF displacing C++ or Java. However, ELF's strengths could help find it a strong developer community.

Figure 2. Application Class Screen

Asked how Applix intends to generate developer interest in ELF, Richard Manly, Director of Product Management and Marketing for Applix, said customers will be doing this in two ways:

> The first is to create a series of SHELF applications which will be offered in Open Source that will show developers what can be achieved by using the SHELF development tools. The first of these applications will be the Linux Palm Desktop (LPD), a graphical desktop application which will enable Palm or Palm Pilot users to download, view and search their PDA data in a familiar user interface. By offering the LPD application in Open Source, developers will be able to use the interface to extend the use of their data into either their own programs or into other applications which run on Linux machines. We'll also offer a link to Applixware for automatic generation of word-processing documents, e-mail and spreadsheets.

Thousands of developers have already used ELF as part of Applixware to build a wide variety of applications. According to Manly, these range from

> using ELF to record and play back keystrokes and mouse clicks to automate often-repeated activities within Applixware to extending the functionality of the spreadsheet to additional analytics to full-scale applications which utilize ELF's ability to integrate with third-party applications using databases (via ODBC), sockets, shared libraries and RPC calls and CORBA via IIOP.

The “Free Stuff!” section of the Applixware for Linux site (see Resources) contains sample ELF applications such as Solitaire, which can be freely downloaded. The Linux Palm Desktop application will be posted at the Applix Open Source Central site. This site will be the focal point for SHELF development including downloading the SHELF distribution and contributed extensions.

A variety of books about using ELF are available from Applix's web site. Both Applixware and SHELF are available for all major Linux distributions as well as Solaris, AIX, HP-UX, Digital UNIX, Irix, Windows 95/98 and NT.

Applix Linux Division

In June, Applix announced a new division that will work with the Linux and Open Source software community to source and brand applications. The Applix Linux Division will also continue to aggressively market, sell and support the company's Applixware product suite for the UNIX and Linux markets.

Applix is currently growing and recruiting additional developers and marketing staff for the new division. Asked how the new division will change Applix, Manly replied,

> By being focused on the Linux market, we'll be more able to respond quickly to the demands and direction of the Linux user base.

Resources

Craig Knudsen (cknudsen@radix.net) lives in Fairfax, VA and telecommutes full-time as a web engineer for ePresence, Inc. of Red Bank, NJ. Craig has been using Linux for both work and play for three years. When he's not working, he and his wife Kim relax with their two Yorkies, Buster and Baloo.

Archive Index Issue Table of Contents

Advanced search

Linux Distributions Comparison

Ellen M. Dahl

Issue #65, September 1999

Distro. summary.

LINUX DISTRIBUTIONS COMPARED

	Caldera OpenLinux 2.2	Debian 2.1	Linux Mandrake 6.1	LinuxPPC 1999, Release 5	Project Independer 6.0-0.2
Price	$49.95	free	$50.00	$32.00	free
Target Consumers	CDEHOV	CDEHOV	EDHOCV	CDEHOV	DEH
Kernel	2.2	2.0.36	2.2.9	2.2.6	2.2.10
Binary Format					
Primary/Other	ELF	ELF libc6; libc5 for compatibility	ELF	ELF	ELF
Hardware Required					

Minimum Install	386, 8, 40	4MB, 100MB HD, 386	i586/50Mhz, 8MB, 350MB	16MB RAM, 100MB disk	250MB dis
Minimum Use	386,8,40 as server; 486,32,300 as workstation	4 MB, 100 MB HD, 386	i586/100Mhz, 16MB, 400MB	64MB/ 350MB disk	16MB RAM
Recommended Use	Pentium, 64, 1000	8MB, 16 w/X, 500MB HD, Pentium	i586/200 Mhz, 64MB, 900MB	1GB (50MB for swap space)	32MB RAM
Packaging					
Package Format	RPM	.deb	RPM	RPM	RPM
Source Code Packages	yes	yes	yes	yes	yes
Management Interface	X, command line, Lizard, COAS, KDE	dselect/apt	kpackage, GnoRPM	X, command line, pdisk	GnoRPM
Group/ Subgroup Packages	yes	classification, 3rd party	yes	yes	yes
Boot Media					
Boot Floppy included	yes	yes	no	no	no

Boot images to select from	module-based, no limit	2 for i386; diff for other architectures	9	n/a	2
Floppies required	none	none from CD; 1 or 2 if CD not bootable; 2 from network	FTP, NFS, HTTP installs,PCMCIA	none	1
Run from					
CD-ROM only	no	yes	yes	yes	no
CD-ROM mostly	yes	yes	yes	no	no
Configuration					
Custom X	yes	yes	yes	yes	yes
Network	yes	yes	yes	yes	yes
SLIP/PPP	yes	yes	no	yes	yes
User/Group	yes	yes	no	yes	yes
Filesystem	yes	yes	no	yes	yes
Printer	yes	yes	yes	yes	yes
Runlevel/init	yes	yes	yes	yes	yes

Install					
PCMCIA Ethernet	yes	yes	yes	yes	yes
PCMCIA CD-ROM	no	yes	yes	yes	yes
from CD-ROM	yes	yes	yes	yes	yes
from Floppy	yes	yes	yes	no	no
from FTP	no	yes	yes	yes	yes
from local filesystem	yes	yes	yes	yes	yes
from NFS	yes	yes	yes	yes	yes
from Tape	no	media only	no	no	no
UMSDOS from Linux	no	no	no	no	no
UMSDOS from DOS	no	no	no	no	no
Media available	yes/CD-ROM	yes	from HTTP or Windows (to -o loop mounted file)	yes	yes, planne
Documentation					
Custom manual included	yes	yes, electronic form	yes	manual is on-line	no
3rd party books/manuals	yes (manuals)	yes	no	yes	no

Extras					
Non-GPLed extras included	BRU, StarOffice, LISA	432 packages available	over 30 apps incl. WP 8.0, SO 5.1, IBM Lotus eSuite DevPack 1.5, QuakeII & Hopkins FBI demos	some MacOS software	N/A
Support					
Included	5-incident e-mail, 5-incdt/90-day install	mailing lists, bug tracking syst, online forums	100-day e-mail	e-mail installation support	none
Optional	support contracts	3rd-party only	nono	none	none
Special certifications	forthcoming Caldera training	-	-	-	-
Sales method used	DFX	DF	DF	DFX	F now, D so

LEGEND	
Target Consumers: C Commercial user **D** Developer/Engineer **E** Desktop/End user **H** Hobbyist/Enthusiast **O** OEM **V** Value Added Reseller	**Sales Method Used: D** Distributor **F** Free via FTP **X** Direct **Hardware Required:** Values are CPU, RAM (RAM for X), Disk

Interview: Caldera's Ransom Love

Marjorie Richardson

Issue #65, September 1999

An in-depth interview with Caldera's number one man.

Ransom H. Love, President and CEO of Caldera Systems, Inc. gave one of the keynotes at the OpenSource Forum in Austin, Texas on June 30. He talked about the impact of Linux and Open Source on businesses today. Later in the day, he graciously gave me some of his time to discuss what we can expect from Caldera now and in the future.

Margie: What are your thoughts about today's conference?

Ransom: It's exciting to see a conference focused on Linux in the enterprise. Just the fact that there is a conference along those lines is quite a vindication, I guess, that Linux is a valid business alternative which is kind of our say. There wasn't a large number of attendees, but oftentimes it's not the quantity but the quality of people looking and the jury is still out on exactly what that means as far as ongoing businesses and relationships.

Margie: Do you think they will go away believing Linux is the answer?

Ransom: Well, you know, as far as the information that has been presented, I think it's been very solid, very focused, very good information. All the feedback I've gotten from everybody in the conference has been very solid. I don't know that anyone is going to be convinced, but I think they are going to walk away

and begin to evaluate. That's all you want to achieve; if they can begin that cycle of evaluating, that's all you need. You just need your foot in the door, because Linux works. That's truly the goal and I think the conference has achieved that. I think attendees were split almost half and half—half had already deployed or were evaluating Linux, the other half were considering it. I think they will begin the process.

Margie: On one of the last panels, one of the speakers was saying that Linux wasn't enterprise-ready at all. You needed to be there to tell him he was wrong.

Ransom: I think the problem is, what does “enterprise-ready” truly mean? Does it mean having a list of features and functionalities that people want? Does it mean it's stable enough? Does it mean it's scalable enough? To say “enterprise-ready” is to say you have given it somebody's definition. To me, it means it works; it does what it says it does. There is no argument whatsoever that Linux does what you say it can do. It's solid; it works; integrators and VARS deploy it as the predominant platform for businesses all over the world. The question becomes, does it have all the fine tuned options, such as SMP, that other systems have? Does it have all the high intricate file systems and other such things? No. But does that mean it's not enterprise ready? No. Because there is an awful lot of functionality that goes into the enterprise, but what you really need is something that works.

Margie: Right. In your discussion, one of the attendees asked if Linux is going to replace NT for the desktop. I liked your answer and I thought you could repeat it briefly here.

Ransom: Okay, very good. I think the answer to that is the desktop itself has changed. It's rapidly evolving from just a PC that is monolithic with a lot of maps and memory, to being broken up. On the software side, the browser itself is becoming the interface, and more and more applications are being served down to the desktop from a server environment using the Internet protocols. If that is where the browser is going, then Linux is going to play a major role. Its small footprint can be remotely managed, it is very stable and all those applications can be fed to the browser as well as any other platform. It's an excellent Java platform. So all of these application servers being developed will feed those types of applications out to a thin Linux environment. And it will play very, very strongly. Even the new devices, the NCs, and such are coming preloaded with Linux. Will it replace that large monolithic PC? Maybe not, but who cares? At that point, Linux will play where it plays well. It will add significant value in that shift, if the desktop, as well as the server, changes form and comes into the more economical, more manageable forms we are starting to see.

Margie: You also talked about standards and certifications. As part of your education courses, you are going to offer certification. Are you doing this in cooperation with LPI, or is this something you are doing independently?

Ransom: No. As you know, we are a platinum sponsor of LPI and played an integral role in getting LPI set up, but they are an independent organization. We want them to be an independent organization, because we want them to supply an industry standard. So they are creating all the certification testing. We aren't even going to touch that. What we will provide is the educational courses necessary to give somebody all the information they need in order to pass that certification. Our courses are not OpenLinux-specific; they are Linux and designed to teach someone how to become certified using the LPI standard. We are working with LPI and contributing, financially and otherwise, to create this certification. Then, in turn, we are creating the coursework that can bring someone up to the knowledge they need to pass those tests. So you get the benefits of all those. LPI certification, open source, you can go through them and do the tests yourself, but if you want some assistance, we can provide you with literally 20 different courses that can get you trained and educated so that you can feel comfortable taking and passing those test suites.

Margie: And they are all Linux in general, not Caldera-specific?

Ransom: Yes, Linux in general. We feel it is critical that the standard be Linux. A Linux-certified engineer means a heck of a lot more than being a Red Hat or Caldera or SuSE engineer. For businesses, for certification of an individual to have any kind of meaning, that certification must have a lot of credibility. So that has been our focus with education from the beginning. We're very excited about the potential of LPI rolling out certification. The other aspect of that is the channels to deliver that education, because many corporations are worldwide, so they need to have educational coursework worldwide. Our announcement last week with IBM where they are going to roll these courses out through all of their educational training centers is a significant and unique announcement in this industry. So we are very excited about the potential of that, because now, corporations and VARs and integrators worldwide can deploy solutions and be able to get people educated, trained and supported. Another aspect of that announcement is IBM is doing worldwide support of OpenLinux—now we have the mechanism to deploy solutions globally.

Margie: I thought that was surprising too, because everybody thought IBM was just working with Red Hat.

Ransom: No, and the relationship with IBM is significant—let me make this clear—not because they are playing favorites. They literally are playing with all Linux providers. But because of our business model and our focus on business,

many aspects of the things we are doing match up on a broader scale for what IBM wants to do. Thus, you'll see more and more relationships with us. Anyone else could step up and offer the same things if they had the same focus and business model as we do. Again, this is just a confirmation of our focusing Linux for business in the match you see now forming with IBM as we roll forward. And you'll probably see that with other companies as well, on that same level.

Margie: We received a good response to our article on standards in the June issue. In particular, your part of it, because you were willing to say more than anyone else. You dominated the conversation, people liked it, and as a result, see you as a leader in the Standards drive. Are you just giving it lip service or do y'all actually have people on the inside working with LSB?

Ransom: Well, Ralf Flaxa, who is actually the head of our engineering team in Germany, is heading up that whole reference platform, which we feel is a critical part of that LSB certification. The reason being, because as you agree to a written specification, you need a proof of concept of that specification—to be sure it actually works and everyone is happy with that specification. You can then do re-integrations on the specification to improve and enhance it. We have freed up Ralf—almost 100 percent of his time is now allocated to working on the Linux Standard Base to help drive, manage and chair it. He isn't the only one working on it, but he is the chair and therefore we are trying to free up his time so that he can, in fact, deliver the reference platform of the standard. In addition, we have and will continue to provide resources to the Linux Standard Base group of committees. For example, for their last meeting we did fly a number of the Debian developers in to attend the meeting, so they could all come together. We are expending resources in other ways to try to facilitate and push and help things along. IBM is also doing some wonderful things, now rallying behind Linux Standard Base and working and collaborating with many other ISVs. They are looking for ISVs to come in and help put additional pressure into this area of standards. We are very excited about that. ISVs are starting to become more vocal about the needs and I think that will help others. Once people realize that ISVs are serious and the applications need to be there—it's just a matter of time and momentum—the pressure will bring about significant movement. One of the reasons why we are so passionate about that is we have worked with VARs from day one. The VAR channel and system integrators have been key, because they have a lot of those applications. We heard about this need a long time ago because of our business focus. That's one of the reasons we have been so vocal and such a strong advocate of standards.

Margie: Exactly what is your Linux tour about? Y'all came by and saw us, is that demonstration pretty much what you are doing everywhere?

Ransom: Well, we actually have three different tours we are doing right now. One is the “I Develop” tour with Oracle. We are the only Linux distribution going worldwide to their “I Develop” conference. So, that's been very positive and has some very good feedback among the developers who are looking to develop and deploy Oracle solutions and related solutions on top of Linux. We are also doing the Network Professional Association tour, where they are touring the country meeting with their affiliate groups and associate groups. They are doing some significant evangelism, if you will, of OpenLinux and education especially. They actually helped us develop our latest administration course that allows someone coming in with an MSVE or CNE background to get the specific training they need to bring them up to speed on Linux. They actually helped us develop that course, so they are out there helping us evangelize it through the Network Professional Association. Our more significant tour is called the OpenLinux tour. Ziff-Davis helped us host it through their conferences and that side of their business. We've coordinated the city-to-city tour ourselves, and IBM and Oracle are co-sponsors. We've gone to eight cities already and we will be going to another seven cities as part of that first phase. That has been very, very successful. We obviously are targeting VARs and system integrators in each of the cities, and we have had an excellent response. The VARs are very enthusiastic about the fact that IBM and Oracle are very serious about Linux. That gives it some validity and our experience with the VAR and integration channels has a very strong appeal. We know what the VARs want and how to help them drive solutions on Linux. So we are pleased with the response.

Margie: When you came by to see us, you showed us Caldera's new easy install, Lizard. Tell us about that.

Ransom: That was when we were actually rolling out our 2.2 product and the first part of Lizard. We will have some exciting announcements coming up here as we get ready to open source it. We don't want to just throw it out there. We literally want it to be an interactive development kind of thing, so that it can literally be a standard in Linux. So we are creating an entire open-source site, so that developers can interact with and maintain and keep it up. A lot of our open-source projects are going to be moving on to that site and into electronic format with more of a developer focus to give the support and the ongoing maintenance of the technology an opportunity to be very successful. There is an announcement that will be coming out soon. So that's what we are doing. It's not just a matter of simply publishing the source; we could do that. We truly want it to be published in a way that is meaningful, so that developers can get the information they need and good access to the technology. We are doing a little more work there.

Margie: Is there a date when this might happen?

Ransom: It's coming very soon.

Margie: Very soon, okay, that will do. How about new features that are as exciting as Lizard? Do you have any of those coming up in the next release?

Ransom: Well, obviously, Lizard was the first integration of the product, so there are a lot of things we didn't have time to put in the first release that we are definitely going to see in the second release. We are very excited about the enhancements we've been able to make to Lizard—additional platforms, other kinds of things we can drive there, so that's a big improvement. We have been able to put a lot of other things into the product—things like unattended install. Lizard is a wonderful thing for a one-time install, but many of our VARs, integrators and the major OEMs want the ability to basically install once and have it replicated across many units. So that's an aspect we are putting into this next product. We have a few more commercial applications, a few more upgrades to existing commercial applications and things like that which will be made available. There are a couple other features, but we are saving them. We think they are really significant from a business perspective, but we don't want to let the cat out of the bag too soon.

Margie: Okay, I understand that. KDE is now a part of your distribution. Do you think the main way to attract people to Linux is to look like Windows?

Ransom: Well, again, we believe in providing Linux as a targeted solution. All of our research on those people who are buying Linux—we have gathered a lot of data—tells us that well over 50 percent of the people buying Linux today are buying Linux and UNIX for the first time. On our city-to-city tour, by the way, a lot of the VARs—well over 61 percent—are novices when it comes to UNIX; they don't even know anything about it. The number of people who are moving from Windows or wanting an alternative to Windows is significant. These people are evaluating Linux. So what we've done with Linux is target those first-time users in such a way as to give them an experience that won't send them running away screaming. They are used to a point-and-click interface and interaction with the system; they are used to it all being graphical; they are used to a kind of WYSIWYG-type environment, so we tried to deliver a solution that would allow them to have a good experience. Now, that doesn't take away any of the power of Linux. Underneath the covers, it's still Linux. In fact, many of these things actually appeal to developers, because they give them a more controlled environment, kind of a single-product environment that they can optimize and play to their heart's content. Now, you'll see us come out with other products very soon that are again more targeted, that have a more WYSIWYG first-time user graphical environment. You will see us come out with a server environment that won't have a graphical environment, but instead will be in a browser. It will be browser-based so that the system can be headless and

keyboardless, and you can deploy and manage Linux completely remotely. So we extend code now to include an entire web interface, not just the graphical KDE, and that's the next phase you will see very shortly. We believe that for businesses and VARs and others to get their hands around Linux, we will do a lot of the packaging and focusing and creating a solution so they can focus on adding to it. They do not have to manage Linux or sort through all these different things to get their solution—we give them a basic solution and they just add theirs to it. What you are seeing is us trying to appeal to the market and the customers who are now moving in and buying Linux for the first time.

Margie: Last year, Caldera split into two parts. Has that worked out?

Ransom: Yes, actually it has. Well, what do you think about 2.2?

Margie: I like it.

Ransom: Has it worked out?

Margie: Okay, I can't argue with that.

Ransom: Actually, it's been very good. It has allowed a lot of focus on the two different areas. There is some significant difference between an embedded-type application and the desktop server or even the non-traditional PC-like devices that are being broken up. There are differences there, and I think allowing us to focus down on the two different areas has been very positive for both sides. The thin-clients group has a wonderful set-top device they've developed; the browser is highly optimized to achieve the environment. It's a wonderful platform. We have been able to focus on delivering real solutions that I think add value, not only to ourselves, but to Linux in general as we publish Lizard and everything back.

Margie: Is your Linux for Business focus working? With your relationships with IBM and Oracle, it certainly looks like it is.

Ransom: Yes, and I think more than ever, the proof is in the pudding. What are you seeing? What type of solutions are we delivering? What type of activities are you seeing in and around what we are delivering? I think the relationships and the products—not just OpenLinux 2.2, but the educational products which have been released and announced, and the many more coming—are the proof in the pudding. We are actually delivering on our promises and commitments and getting solutions to the market. People worldwide are recognizing the value of OpenLinux. It's totally different from any other Linux out there. There really is no comparison, and yet we are using the same kernels and libraries—do you see what I'm saying? It is the focus that makes the difference. I think that will

pan out even more in the coming days and weeks and months as more and more announcements come out.

Margie: So are you seeing an increase in market share?

Ransom: Oh, my goodness! It's fun!

Margie: I'll take that as a yes.

Ransom: Well, again, I don't know that we are taking market share from anyone, but we are definitely taking the new users who we are targeting. A lot of them are coming over and taking a look and evaluating. We are definitely appealing to the VARs and systems integrators worldwide, who we are also targeting. That's our customer. It's not that it's at the expense of anyone else—I guess it is in some degree, because we are taking a larger percentage of the new consumer of Linux.

Margie: Tell us a bit about Caldera's business philosophy.

Ransom: Well, to give you an idea, we kind of believe the kernel and the underlying infrastructure needs to be and should be open source—there is no question about it. It adds so much value to the application providers, the solution providers, everyone, because having an open source allows them to optimize, customise and deploy solutions more effectively. Where we differ from some is that we believe there is a role and place for binary-only applications and solutions or proprietary hardware components—a very important role. Billions of dollars are spent, you know. We shouldn't run away screaming, saying that is bad, that is taboo. Because there is significant value in a solution that the combination has put together to solve a business need that you may or may not get from a pure open-source model. It may be years; how do you provide incentives for people to work on some aspects that just aren't appealing? So there is an element and a need for both proprietary and open-source software. The models should be used where they make sense, and the combination which creates the best solution for the customer, the best solution for the industry and everybody involved. That's where we vary a little bit from some. We feel very strongly that to solve business solutions, to solve the customer needs, the best of both worlds is really the right approach.

2.2 is a prime example; we have Power Quest and PartitionMagic, which give a non-destructive, on-the-fly resizing. Well, we don't have those partitioning tools yet in an open-source fashion that give you the same level of confidence with the same level of interface. Why not take Linux with all of its beauty and functionality and couple it with this? Now, you've created a real solution that a greater majority of people can use today. So the combination is good.

Another example is Apache/IBM; you have the security aspects of that which will never be open source. Is that bad? No! The whole nature of the reason it's not is that it's secure! You can't open source it. So, is that wrong? Is that bad? No. It has a value, and the combination of the two make perfect sense as a business opportunity and solution. I'll fight with the best of them on protecting open standards and maintaining the basic kernel core as open source, because that's the value of the whole model. But I think we need to develop interfaces that allow for binary interaction—that it's not just open source and forcing open source. Another example: we have all these major players coming in, and we have meetings with nearly all of them. They are very excited about trying to help Linux move forward. They are looking at publishing technologies that would take years to develop, just as a good gesture to contribute back to the community and add value. They want Linux to succeed. But you know what? There are some aspects they will never publish, nor do they want to, nor does it make sense for them to do so. Is that bad? No; you get the benefit of all this technology and all the knowledge, and the customer gets the solutions because they are all integrated. That's great; let's evaluate each one and not throw up official barriers.

Margie: Will Caldera be going public?

Ransom: I think it's everybody's dream to go public, especially in the software industry. Frankly, my major goal for our company is to be a valid, viable business and let the natural consequences take place from that. There is no question that we, like every software company on the planet, are preparing ourselves for the eventuality of doing something bigger, grander and better. Our model is a little different than others; it's not a rush to get out and be the first one to do an IPO. That's not the issue. When we do an IPO, we want it to be a solid business decision with solid business solutions surrounding it that can sustain an IPO. I guess we are a little bit more conservative in some ways in that we are going to take the time to forge the relationships, to build the business, to deliver the solution, so that when we do an IPO, we are doing it to fund the business and not to buy one.

I think there are many companies preparing themselves for IPOs. I think that is great. I won't lie to you; we will do everything in our power, but we'll do it when it's right for the business and we feel we have a sustainable business moving forward. And we do now, as far as the numbers, we are looking fine, but we are in a phase of maturation. It's important to see what we look like after we go through puberty!

Margie: Sounds good. Last question: what did you have for breakfast this morning?

Ransom: Fruit and oatmeal.

Margie: All right—my favorites. Thank you for your time.

Archive Index Issue Table of Contents

Advanced search

Multicast: From Theory to Practice

Juan-Mariano de Goyeneche

Issue #65, September 1999

Broadcasting over the Internet—a look at developing applications for this new technology.

As the Internet grows up, new communication needs arise. First, e-mail and FTP were enough for most people. Then the WWW arrived and people wanted to see graphics, not just plaintext. Now, even static graphics are not enough; real-time video and audio are demanded.

As communication needs evolve, communication paradigms originally designed to deal with e-mail and FTP need to evolve too. A new one that has developed is “multicast”.

The Problem

Imagine transmitting an event over the Internet (perhaps a Linus Torvalds conference), and multicast is not available. A single source of information, which could be a computer connected to both the Internet and the video cameras and microphones Linus is talking to, is transmitting multimedia streams to several hosts dispersed over the Internet. Of course, traffic should be sent as efficiently as possible—the less bandwidth used, the better.

With pre-multicast technology, two communication paradigms are available, both of which are inadequate.

The first one is *Unicast*. TELNET, FTP, SMTP and HTTP are unicast-based protocols with *one* source and *one* destination. To send to multiple destinations, different communication paths are needed between the source and each of the destinations (see Figure 1.a). Therefore, a copy of each audio and video stream would need to be made and sent separately to each receiver. Clearly, this is not affordable. Even if you are quick enough in copying real-time audio and video streams, both your network and the Internet would collapse.

Audio and video are extremely bandwidth expensive. Obviously, TCP cannot be used in multicast applications, as it is clearly unicast-oriented.

Figure 1.

The second choice is *Broadcast*. The broadcast paradigm (see Figure 1.b) saves a lot of bandwidth compared to unicast. If you want to send something to all computers on your LAN, you don't need a separate copy for each. On the contrary, only one copy is sent to the wire, and all computers connected to it receive the copy. This solution is better for our problem but is still insufficient, as we probably need to broadcast to only some of our computers, not all. Even worse, it is almost certain that many hosts interested in your conference will be outside your LAN. While broadcast performs well inside a LAN, problems arise when broadcast packets are routed across different LANs. Thus, broadcast is good for applications and protocols that don't need to cross LAN limits (such as ARP, BOOTP, DHCP and even routed), but it is not good enough for our problem. Finally, it is very likely people want to have more than one video conference at a time, when only one broadcast address is available.

The Solution: Multicast

After having looked at the problem, it is apparent we need a solution that provides the following:

- Allows data to be sent to multiple receivers in an efficient way, avoiding per-receiver copies.
- Is not constrained by arbitrary network limits, so it can reach anyone, anywhere on the Internet.
- Differentiates between multiple and unrelated transmissions, so that a computer may know which ones are of interest for applications.

The third point relates well to radio or TV channels (not cable TV). If you are interested in a particular channel, you tune your receiver to listen to a particular range of frequencies and discard the rest.

The solution that meets all three requirements is *multicast*. Figure 1.c shows that host 1 sends data only *once* (i.e., no per-receiver copies are made) and only hosts interested in this data (hosts 3, 5 and 6) receive it.

Multicast Addresses

The radio/TV example will remain a good starting point for the rest of the article. In the same way that multiple frequencies ease the process of recognizing and isolating different TV channels, multiple multicast addresses ease the process of recognizing the multicast traffic of interest.

The range of IP addresses is divided into classes, based on the higher order bits of the IP address. Multicast addresses are class D addresses: those starting with the first three bits set to 1 and the fourth set to 0. This means multicast addresses range from 224.0.0.0 to 239.255.255.255. This is the range of "frequencies" in which you can transmit or listen for traffic. Each "frequency" identifies a different and specific multicast group.

Some of these multicast addresses are well-known, reserved for a specific purpose. For instance, 224.0.0.1 is the *all-hosts* group. Just "ping" this address, and all multicast-capable hosts on the network should answer. Any multicast-capable host must join this group at start-up on all its multicast capable interfaces. 224.0.0.2 is the *all-routers* group, and so on. In any case, your multicast applications should never send datagrams to addresses 224.0.0.0 through 224.0.0.255, as they won't be forwarded across multicast routers. Similarly, you should avoid groups 239.0.0.0 through 239.255.255.255 as they are reserved for *administrative scoping*. See the "Multicast over TCP/IP HOWTO" (included in the Linux Documentation Project) for further details.

Configuring Your GNU/Linux Multicast Box

In order to play with multicast, your GNU/Linux box needs special configuration. Your kernel must be compiled with **IP: multicasting** enabled. This will add support for the IGMP protocol (Internet group management protocol) to send and receive multicast traffic. If you keep on playing with multicast, it is quite likely you will need to use your box as a multicast router, as old routers do not support multicasting. In that case, check the HOWTO for several additional compile options which must be enabled (i.e., say **YES**). You will also need the **mrouted** application, a daemon which instructs the kernel on how to forward multicast datagrams when acting as a multicast router (mrouter).

Finally, you need to set a default route for outgoing multicast datagrams. Assuming the eth0 network interface is to act as that outgoing route (your application can instruct the kernel to send its datagrams using a different interface if needed), you'll need to use:

```
route add -net 224.0.0.0 netmask 240.0.0.0 dev eth0
```

Writing a Complete Multicast Application

Now that multicast is defined and your hosts are set up, I will explain how to write multicast applications while developing one. Its aim is to be both a didactic and useful tool. The reader needs a basic background in network programming using the sockets API. *UNIX Network Programming* by W. Richard Stevens, *Internetworking with TCP/IP Vol. 3* by Douglas E. Comer and the **setsockopt** man page are helpful references.

The idea for the application in Listing 1 came from a popular TV commercial in Spain: a little boy takes his father's mobile telephone, starts calling numbers randomly and saying: “Hi, I'm Edu. Merry Christmas!” His father gulps when he discovers it and, of course, the lesson is how cheap this company's mobile phone calls are (in Europe, local calls are quite expensive).

Listing 1.

Our program (see Listing 1) will do the same thing: it will send to the multicast group and port, passed as command-line arguments, the string “Hi, I'm *name_of_machine*. Merry Christmas!” along with the time to live (TTL) of the message. The program is short and simple, but it is also quite useful. I have used it several times when configuring multicast networks. You can run it on all your machines to see whether they are sending and/or receiving traffic. The TTL is very handy when using multicast routers and/or tunnels, as it makes it easy to determine the lowest TTL needed to reach a given destination.

The first lines of the program are the usual **include** statements. I tried to add comments to point out which functions and/or data structures need them. In the main function, variable definition and basic initializations are done in lines 27 to 44. Later, we use a dedicated socket for sending **(send_s)** and another for receiving **(recv_s)**. These sockets must be **SOCK_DGRAM** (UDP), as TCP does not support the multicast paradigm.

Sending Traffic

When multicast was implemented, the sockets layer was extended a bit to support it. That support came via the **setsockopt/getsockopt** system calls.

Three of the five new *optnames* (see the setsockopt man page) were intended for use when sending data: **IP_MULTICAST_LOOP**, **IP_MULTICAST_TTL** and **IP_MULTICAST_IF**. They are all at the IPPROTO_IP level.

If **IP_MULTICAST_LOOP** is set, all multicast packets sent from this socket will be looped back internally by the kernel. This way, the rest of the applications waiting to receive traffic for this group will see it just as if it had been received by the network card. We are not interested in that behavior for our application, so it is disabled in lines 65 to 69. By default, loopback is enabled.

The TTL field of the IP header plays a primary role in multicasting. Its original role of avoiding problems with packets being looped forever due to routing errors is kept, but a new one is added: that field is also associated with a meaning of “threshold”. It acts as a delimiter to keep multicast packets from being forwarded without control across the Internet. You can establish frontiers by specifying a multicast packet will cross your multicast router only if its TTL

field is greater than a particular value. This way, you can multicast a conference restricting its scope to your LAN (TTL of 1), your local site (TTL<32), your country (TTL<64) or allow it to be unrestricted in scope (TTL<256). Our test program lets you specify the TTL on the command line, then sets it using the **IP_ MULTICAST_TTL** option. If none is specified, TTL 1 is assumed (see lines 52 to 62). If you are using multicast tunnels or your applications are separated by multicast routers, you can run the program on both ends by increasing the value of the TTL field until the two programs "see" each other. This way, you can easily discover the minimum TTL necessary for your applications to communicate.

If not otherwise specified, outgoing multicast datagrams are sent following the default multicast route set by the system administrator. If this is not what you want, you can specify another output interface for that socket. Our sample program is quite simple and does not need this feature, so we did not use the **IP_MULTICAST_IF** option. Instead, we let the kernel choose the correct route. If you need it, write code such as:

```
struct in_addr interface_addr;
setsockopt (socket, IPPROTO_IP, IP_MULTICAST_IF,
   &interface_addr, sizeof(interface_addr));
```

filling the **interface_addr** structure with a suitable value. If later you want to revert to the original behavior, just call **setsockopt** again using **INADDR_ANY** as the interface field.

Receiving Traffic

Your radio or TV must be tuned to receive the channel you want to hear. In a similar way, you must "tune" your kernel so that it knows which multicast groups are of interest. This is known as "subscribing" the host to a particular group. Note it is the host, not the process, that is subscribed. Processes are bound using **bind** to a particular multicast group/port pair and must tell the kernel they want to receive traffic for that group. The kernel then knows it must not drop packets for that group. When it receives them, it makes copies for all processes bound to that multicast address and port pair. When the last process that remains subscribed to the group "drops membership", the kernel stops sending these packets to the upper layer protocols and ignores them again.

In short, if you want to receive traffic from a multicast group, you must take the following steps:

- Create the socket (lines 71 to 74).
- Bind the group/port (lines 81 to 84).

- Optionally, use the **SO_REUSEADDR** option (lines 76 to 79), so that more than one process can bind the same group and port on the same machine, i.e., have multiple receivers.
- Join the group (lines 87 to 92).

The **IP_ADD_MEMBERSHIP** option expects a pointer to a **struct ip_mreq**. This structure is defined in netinet/in.h. The first field, **imr_multiaddr**, contains the group address you want to join. The second, **imr_interface**, holds the IP address of the interface to which the group will be joined. This is a key point: membership is associated with both groups *and* interfaces. You do not just join a group; you join a group *on* a network interface. If your host is multi-homed, you can join the same group on all your network interfaces, on one of them or even on some of them. This way, the application will get packets sent for that group *and* received on that particular interface.

Normally, you want to receive traffic for that group and you don't care which interface received it. In those cases, fill the **imr_interface** field with the **INADDR_ANY** wild card (see line 88).

When you are done, you might want to drop membership (stop being a member of that group), although this is not strictly necessary if you are going to close the socket right afterward. The kernel will drop membership for you on all groups the socket was subscribed to when you close it.

If your process drops membership for a particular group but keeps the socket bound, it will keep receiving that group's traffic as long as any other process in the host remains a member. Joining a multicast group only tells the IP and data link layers (which in some cases explicitly tells the hardware) to accept multicast datagrams destined to that group; it is not a *per-process* membership, but a *per-host* membership.

The rest is easy; we fork and let the parent send messages (lines 123 to 137) and the child receive them (lines 104 to 122). As we told it not to loop back, we do not see our own messages. Change the **IP_MULTICAST_LOOP** option, and you'll find you are talking to yourself.

Conclusion

Feel free to test, modify and enhance this example program. You'll probably see that there are certain subtleties not fully addressed in the text. It is difficult to cover everything in a short article, but you can check and complete it by reading the Multicast HOWTO (tldp.org/HOWTO/Multicast-HOWTO.html).

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue65/3041.tgz.

Juan-Mariano de Goyeneche (jmseyas@dit.upm.es) moved to GNU/Linux quickly when he realized that it was much easier to debug and modify programs when one has the sources. While he finishes his educational career, he collaborates with the Telematic Systems Department (DIT) at UPM, Spain, working with CSCW multicast applications. He is the author of the “Multicast over TCP/IP HOWTO”.

Archive Index Issue Table of Contents

Advanced search

The 19th Century Meets the 21st

Paul Murphy

Issue #65, September 1999

Mr. Murphy describes how he set up frame-relay service for the old Brooklyn apartment where he lives.

I live at the foot of the Brooklyn Bridge, in a 150-year-old building that used to be the headquarters of the Brooklyn Railroad. Before the bridge was built, barges and ferries docked along the piers. Trains brought people and goods into Brooklyn and beyond. The bridge killed the ferries, the railroad and, eventually, the neighborhood—the heart of Brooklyn in Walt Whitman's day.

After a century of decline, the neighborhood, Fulton Ferry Landing, is being reclaimed by artists and people like me who work anywhere, thanks to the Internet. Disconnected for a hundred years, due to a network of roads that ignored it, Fulton Ferry Landing changed little. Today, it is quickly being reconnected to the rest of the planet, as a result of the most efficient transportation network ever devised.

Figure 1. View of Brooklyn Bridge from Apartment Window

Wall Street—just across the river—is one of the most *wired* neighborhoods on the planet. The rest of New York, however, lags behind most of North America in terms of connectivity options: few cable modems, little DSL, and ISDN being more expensive than digital leased lines. Most people make do with a modem. After seven years of dial-up access, I decided I could no longer wait for the dissolution of our local telephone and cable monopolies—it was taking too long.

Thanks to Linux, some volunteer work and a bit of luck, the old headquarters of the Brooklyn Railroad is now one of the most wired old buildings on the planet. Each apartment has two data ports connected to a router in the basement. The router is connected to the Internet through a digital leased line. We now have high-speed connectivity, 24 hours a day, for less than it used to cost us to maintain dial-up accounts.

Why Wire?

A few months ago, I asked the building's residents to let me wire the building with CAT5 in order to set up a building network, because it made sense both economically and organizationally. Sharing resources, after all, is the whole point of packet switched networking—*not* wiring an apartment building is perverse.

Three years ago, I used to walk into many businesses that had each desktop computer connected to the Internet through a dial-up connection. Today, most of those businesses have connected their entire intranet to the Internet though a single, large pipe. Dial-up connections are expensive and inefficient; no IT

organization would dream of setting things up that way. Apartment building dwellers, however, have barely begun to question the way they approach the problem. Although they enjoy high-speed, permanent connectivity at work, they don't question the AOL dial-up ritual at home.

Costs

To date, no one is offering residential network management service. If you want a building network, you will need an on-site expert to set up and maintain it.

In the U.S., the local telephone company is responsible for wires to the building. Residents are responsible for wires in their apartment. The building owner is responsible for the wiring within the rest of the building. I suspect it will take at least another two to three years for people to realize that IP is as fundamental as telephone service. At that point, they will start making noise about wiring capable of carrying data from the basement to their apartments. Network equipment vendors will start building and pricing hardware for this market, and of course, residential network management companies will be formed. In the meantime, we have to plan and build everything ourselves.

My building was not prepared to provide a network infrastructure. I guessed that trying to convince a majority of the shareholders to do so would be a waste of time, so I offered to pay for it all and resell the service to anyone who wanted it. Everyone with a computer has joined. While I would have preferred not to absorb all up-front costs, I have enjoyed the privilege of making decisions without running them by a committee.

The most significant up-front costs are the wiring, the router and the computer providing name, mail and web services. Running CAT5 wires from each apartment to a central hub—in our case, the basement—is never going to be cheap. In most cases, however, it will cost far less than it cost me. The age of the building worked against me.

In the early 20th century, services were run as if they would never require replacement. Electrical wires were buried in plaster walls. Telephones were wired directly to the building's exterior. Telephone jacks were a 1950s innovation, an early example of plug-and-play. Today, architects frequently design electrical systems to be accessible without the help of a demolition crew. Those who are truly forward-thinking will design easily accessible, parallel conduits: one for electricity and one for data. Today, “data” usually consists of telephone and cable television wires. Tomorrow, those two will be joined by computer network wires, which soon enough will also carry telephone and television data.

In our building, nothing is straightforward. Throughout the years, conduits have been run through the wood and concrete floors to carry electrical, telephone, intercom and cable TV wiring. None were large enough to accommodate additional wires. Running a new conduit was estimated to cost almost $1000 per apartment. That expense was impossible to justify at that time.

Figure 2. View of New York from Apartment Window

While I was mulling over what to do about this network wiring problem, another arose. The building ran out of telephone wires. Whoever did the capacity planning when the central wires were installed never considered fax lines, dial-up lines and two or three voice lines per unit. Also, the wires were old—many broke due to corrosion and many were static-filled. Clearly, I had another project on my hands.

Actually, I was lucky the building reached the end of its telephone network lifetime when it did. Any earlier, and I would not have had the foresight to run network lines in parallel with the new telephone lines. Any later, and I would probably have invested in a high-speed solution for myself and would not even have considered doing the work on a communal scale. The incremental cost of running the network wires was negligible, so I decided to go ahead and do it.

The great irony, of course, is that everyone has now canceled the lines they had for their dial-up service. Under the old system, we would now have plenty of lines.

Besides the wiring, the router and central computer turned out to be the other big cost in this sort of operation.

The router is expensive because each apartment needs its own subnet. I asked Cisco what they sold that could do the job. They literally answered that buying a router from them would cost me “both arms and both legs”. They did, however, suggest a “cheap” alternative: a low-end router and a switch, a solution that would have cost me about $3,500. I was not willing to spend half that much to solve this piece of the puzzle. I was fairly sure I could build what I needed using Linux.

My neighbor, a Linux guru, assured me I could. Before long, he and I had done the research and mapped out a strategy that worked.

The hardware we needed was free. Businesses all over town have mountains of 486s gathering dust in their storerooms. They are *thrilled* to give them away! As you know, the operating system we decided to use was also freely available.

Since our router was going to be a general purpose computer, we decided to run all of the shared services on the same computer. This simplified a lot of management issues. It also made disaster recovery relatively straightforward. We built a second, identical machine that can be swapped in for the first at a moment's notice. This sort of approach is practical only if a single machine is involved.

Figure 3. Apartment Building

Architecture

At the start of the project, I had one overriding goal: keep the architecture as simple as possible. I could not guarantee a networking wizard would be available when things failed. In fact, our backup system administrator is a 12-

year old resident who knows little about computers; I figured she would be easier to train than most adults. Knowing I was going to have to write thorough documentation about everything I implemented helped me stick to my goal.

We knew Linux could support multiple Ethernet interfaces. We were not sure where to find a card with Linux drivers that could interface with our DSU. A bit of Net research turned up a Canadian vendor, Sangoma Technologies, that seemed to be selling exactly what we needed. Five minutes on the phone with one of their Linux guys convinced me their WAN pipe product would do the job. At $550, it was the most expensive piece of hardware I had to buy, and it certainly beat Cisco's “cheap” solution.

I now had all the pieces: a frame-relay line from the outside, a DSU, a router, a hub, a general purpose computer, wires and a willing alpha tester. I just had to work out the details.

Network Topology

We originally planned to isolate each apartment behind an Ethernet interface. Of course, that seemed ridiculous for those with a single Windows 95 box. We then considered putting all the single machine apartments on their own segment. This presented an evolutionary problem. Whenever anyone bought a second machine, we would have to change IP addresses, physical connectivity, etc. We were stuck between over- and under-engineering the network, until my neighbor remembered some work he'd done earlier for a client in Atlanta.

He remembered Linux supports something called Ethernet aliasing. This allows a single interface to support multiple networks. For example, a single Ethernet card can be configured to support ten apartments, each of which is assigned its own subnet. This turned out to be the perfect compromise. We could logically isolate each apartment without having to use many Ethernet cards and several computers.

If an apartment grows into needing more thorough isolation, we can *upgrade* it to its own Ethernet board! By the time all available slots are used in our current 486, it will have to be replaced in order to deal with the Y2K issue. By then, maybe the router vendors will be selling solutions with more down-to-earth prices.

Security

When I first began discussing the network idea with other residents, security seemed to be at the top of their list of concerns.

We worked out a few security schemes using proxy and masquerading facilities. Whatever we ultimately decided to do had to be configurable on an interface-by-interface basis. I personally wanted access to my computers from the outside world. Luckily, Linux supports that sort of granular security.

One day, I happened to mention the various options to a relatively computer-savvy neighbor who runs a local area network in her apartment. She was horrified that I would consider implementing a security scheme at the building level. She wanted control over her own security so that she could access her machines from anywhere on the Net. After a bit of discussion, we realized the original requests for high security were all from people who used Windows 95 to dial up through AOL.

It turns out the concerns were the result of alarmist articles in the local papers —security threat articles fail to put the subject in perspective. The least savvy are most easily frightened, even though they are least at risk since they use operating systems with few services that can be abused.

Having come to that realization and remembering our “keep it simple” goal, we decided to leave security up to the individual apartment. After all, AOL does not provide any special security to the lone PC connecting through its network.

Name Services

We toyed with the idea of allowing everyone to register their own domains, but finally decided against it as this would have created too much work. Instead, we registered a domain for our building, 8OldFulton.com, which is related to our physical address. This is one of the few cases in which I think geographic addressing of any kind makes sense. Given the choices we made, the administrative burden of adding a machine or cluster of machines is relatively light.

Mail Service

Mail service is not yet settled. At the moment, we run a POP3 server, because it is essentially administration-free. POP3 is not, however, particularly friendly for people who travel a lot or use multiple computers. Therefore, it is very likely I will eventually bring up an IMAP4 or web-based mail server.

Anyone who wants a more flexible e-mail system immediately in place needs to set up and maintain their own.

Web Service

It is tempting to offer web hosting services for everyone in the building. This would, however, run counter to our “keep it simple” goal. Although there is little complicated about allowing people to set up and maintain home pages, the peripheral support involved is potentially significant. As people become more sophisticated and web development and management software becomes easier to use, my policy will probably change.

Currently, the web server we run serves only private building information: contact information, bylaws, house rules, meeting minutes, etc. I am sure some public information (e.g., directions) will eventually find its way onto the server.

As with special mail servers, anyone wanting to run their own web server is free to do so, on their network segment.

Figure 4. Apartment Building Showing Proximity of Bridge

Implementation Issues

I find it hard to quantify the difficulty involved in setting up the network. My neighbor and I both have done quite a bit of UNIX system administration. Tasks that seem easy to us, like configuring sendmail or name service, might require quite a bit more effort for a beginner. Luckily, the Linux community is extremely supportive. Before embarking on a project like this one, anyone unfamiliar with

system administration should make sure they know how to deal with the following issues:

- kernel recompilation (for WAN pipe support)
- interface configuration
- routing
- name service
- sendmail
- HTTP server setup

We found the most difficult task was setting up the WAN pipe. Because it is not a common router, the telephone company and ISP tend to blame it for every problem—Sangoma is used to this. They ship excellent debugging tools with their hardware, and their installation support personnel are top notch.

The Future

Having the network in place is a great first step. We now have something very solid to build on. Immediately, we all had better Internet access.

We are currently evaluating the purchase of a RaQ from Cobalt Networks. It would provide a more flexible e-mail system and would allow each apartment to maintain its own web site. Under the hood, the RaQ runs on Linux, of course!

Within a few months, I suspect most people will have given up their fax lines. They were often justified because they were shared with the computer. Now that they are stand alone, it probably makes more sense to use the JFax or efax services. It is cheaper (JFax) or free (efax), and more flexible than a dedicated phone line.

When we can buy IP telephones that look and act like telephones, we will buy them. I can easily imagine this building buying no local lines from Bell Atlantic within five years. Between IP telephones and the incredible calling plans offered by our national cellular providers, local lines might not make any sense.

Installing a building security camera will now cost us about $800—the price of an IP camera.

We will likely bump the network up from 10BASE-T to 100BASE-T within the next two years. I suspect a gigabit network will become necessary once we all start using net-based video broadcasts. If that turns out to be impossible over copper, we will run fiber through the old telephone wire conduits. The wire was left in place so that it would be easy to pull the fiber.

Acknowledgments

Paul Murphy spent almost ten years writing software on Wall Street. Today he is a technical partner at Brushfire, a venture capital firm he helped found in 1997. He has advocated free software throughout his career, much to the dismay of his employers. In his spare time he rides motorcycles, plays the violin and raises trouble-free children. He can be reached at murphy@brushfire.com.

Archive Index Issue Table of Contents

Advanced search

Supporting Multiple Kernel Versions

Tony Wildish

Issue #65, September 1999

Expect scripts to help you support multiple versions of the kernel across different platforms.

I work in the Atlas experiment at CERN. Many of the groups in Atlas are beginning to turn to Linux as the operating system for the next generation of particle-physics experiments. Among the teams working on the data-acquisition, there is often a need for specific versions of the Linux kernel. Typically, the teams are using special PC cards, such as ATM cards, where the drivers may not yet be available for all kernel versions, or for which patches must first be applied to the kernel. Both SMP and uniprocessor machines are used, and team members want the same kernel with the same patches for both flavors. They also wish to share software and meaningfully compare results.

This article describes how my team supports this need. I will assume you are familiar with the basic process of configuring and compiling a kernel.

We needed an environment in which a range of kernels could be configured, built, packaged for distribution and later installed in a coherent and consistent manner. The result is our “kernel repository”, containing tar files of the source code for several kernel versions with different patches applied, tar files of the compiled kernels, and a set of scripts used to compile and install the kernels. We wanted to ensure that the configure and build steps were fully recorded and any kernel configuration could be reproduced at a later date, even if the details of the build environment had changed. Also, we wanted to keep up with newer kernel versions (at the time, 2.2.0 was about to be released), so we tried to make it easy to add new versions as they came out. We also wanted the distributions to be easy to install, so that people could use them without knowing the details.

Availability

Our kernel repository and all its associated tools are accessible on the WWW at www.cern.ch/Atlas/project/kernels/www/kernels.html. The kernels are available as separate distributions of precompiled binaries and source code. A technical note is included, which goes into greater detail on some points and helped form the basis of this article.

Preparing the Source Distributions

The original kernel sources were downloaded from the Web. Each kernel unpacks into a single subdirectory—/linux. Since users may want several kernel source trees available at the same time, we rename this directory to /linux-*kernel_version_number.orig* and repack the tree using **tar** and **bzip2**. To give a clear example, if I had downloaded the tar file for version 2.2.0 of the kernel, I would repack it with these commands:

```
cat linux-2.2.0.tar.bz2 | bzip2 -d | tar xf -
mv linux linux-2.2.0.orig
tar cf - linux-2.2.0.orig | bzip2 >linux-2.2.0.orig.tar.bz2
```

The kernel repository includes scripts which will fetch and repack kernels for you.

Whenever any patches were applied, the corresponding source tree was renamed to reflect the patch and sometimes the version of the patch. For example, 2.0.36 with the “bigphysarea” patch is packed as linux-2.0.36.bphys.tar.bz2.

Configuring the Kernels

To configure the kernels, I wrote an Expect script called KernelConfig.exp. Expect is a tool for automating interactive processes (see “Automating Tasks with Expect” by Vinnie Saladino, *Linux Journal*, October 1998), and it is ideally suited to this task. KernelConfig.exp runs **make config** at the top of the kernel source tree and answers the questions for you. The beauty of controlling the configuration by an Expect script is that it is insensitive to the kernel version used with it. This script should be able to configure any Linux kernel version. I have run it on all stable kernels from 2.0.33 to 2.2.7 and on a number of the 2.1-series kernels. While it may not give an optimal configuration for any kernel (whatever that might mean), it does provide consistent and reproducible configurations.

Some configuration options are hardwired in the script, such as:

- Support for EXT2 and Minix file systems for compatibility with the majority of the available rescue disks.
- Support for RAM disk and initial RAM disks. This is essential because the kernel uses modules for most of its functionality and needs the initial RAM disk support to boot on the widest possible range of hardware.
- Compile the kernel for a Pentium-class processor—we have no 386 or 486 machines.
- Support for NFS and for root-on-NFS. Eventually, booting over the network might be useful, in which case this option is mandatory.
- Enable the “experimental code” option. This might seem dangerous, but it is needed to include the ATM code.
- Module-versioning is disabled. While this means one less safety net for the user, it does make life easier if we have to move modules around.
- Support for all classes of network cards, SCSI cards, sound cards and other classes of hardware is enabled. The individual card support is compiled as modules.

Some hardware support is plainly excluded, even though it is available as modules, because at least one of the kernels in the range we are using does not compile. Such problems are normally caught and fixed rapidly, but if the software is not important to us, I leave it disabled for consistency. This happens for a few sound cards, the infrared devices (broken in 2.2.6) and at least one ATM card we don't use. This list may grow as more kernels are produced, so you should look at the configuration log or the script to see what it does. For all other options, the script will select to compile the code as a module if possible; otherwise, it will use the default offered by the configuration. The output of the configuration is stored in the KernelConfig.log file in the current directory. If this file already exists, the output will be appended to it, not written over existing information.

Two words of caution are necessary. The script reacts to the defaults and will therefore react differently if the default changes. For some code, the default changes over time. Something available only as built into or excluded from the kernel in the 2.0.*x* series might be available as a module in the 2.2.*x* series. In this case, it will be built in or excluded from the 2.0.*x* series, according to the default, but will certainly be built as a module for the 2.2.x series. If you need that feature at boot time, be warned.

Secondly, the defaults are taken from the .config file if it exists, or from the file arch/i386/defconfig if the sources have never been configured. If you configure the kernel by hand and set some options before running KernelConfig.exp, it

will accept those settings as default. For truly consistent results, run **make mrproper** before running KernelConfig.exp.

SMP support is tricky. In the 2.0.*x* series, SMP support had to be enabled by editing the Makefile or by building the kernel with the command **make SMP=1 bzImage** or a similar one. In the 2.2.*x* series, SMP support is a configuration-time option, and the Makefile no longer needs to be changed. KernelConfig.exp enables or disables SMP support, depending on the value of a user-defined environment variable **SMP_SUPPORT**. If this variable is not defined or is empty, the script will not enable SMP support. If it is non-empty, the script will enable SMP support in the 2.2.*x* series.

This is not enough for the 2.0.*x* series, where the value of the make-macro **SMP** must be true when the kernel is compiled, not when it is configured. I get around this by defining **SMP_SUPPORT** to have the value **SMP=1**. I can then run KernelConfig to configure the kernel, and **make $SMP_SUPPORT bzImage** afterwards. For the 2.0.*x* series kernels, the *value* of **SMP_SUPPORT** ensures that the kernel is built with SMP enabled at compilation time. For the 2.2.*x* series, the very fact that the variable is defined causes KernelConfig.exp to enable SMP support at configuration time. This gives a consistent approach to SMP for both the 2.0.*x* and 2.2.*x* series kernels.

Building the Kernels

I use a Bash script called KernelBuild.sh to compile the kernels and produce the binary distributions. It takes one argument, the name of a kernel source file (without the ".tar.bz2" extension—e.g., **./KernelBuild.sh linux-2.0.36.bphys**). It starts by defining a few environment variables:

- **MY_WORK** is my working directory. Here, the kernel sources will be unpacked and the distributions will be built. KernelBuild.sh expects to find all the scripts and tools it needs in the directory $MY_WORK/bin. It will also ensure the correct subdirectory structure exists for building the distributions. The built kernels are installed here, not under the root directory, and the distribution tar file is created at this level. Users can unpack it in the true root directory of their client machines.
- **MY_SRC** is the directory containing the prepared kernel sources. I set this as a separate variable to allow customisation of the compilation scripts in another working directory, without having to copy the sources with them. For example, separate teams who wish to build their own versions of the

kernels could set **MY_SRC** to the source directory in a common repository and use the sources directly from that location.

- **MY_ROOT** is the root directory from which the link is set to the source code of the kernel being compiled. In other words, KernelBuild.sh sets a soft link from $MY_ROOT/linux to point to $MY_WORK/linux-2.0.36.bphys—or whichever version it is compiling. In a normal Linux environment, **MY_ROOT** would be /usr/src, and the link would be set from /usr/src/linux to the actual source tree. By allowing it to be another directory, it is possible to compile the kernel as a normal user in another directory instead of working as root. You set the link /usr/src/linux to point to $MY_ROOT/linux once, as root. Then you can change the link $MY_ROOT/linux, as the user who compiles the kernels, as often as you wish.

- **KERNEL_VERSION** is simply the input argument.

The **MY_*** environment variables may be defined externally if you wish, and will not be overridden by the script. **KERNEL_VERSION** will always be set from the input argument.

KernelBuild.sh does not actually do the work of compiling the kernels. For this, it uses two other scripts, Meanwhile.pl and KernelBuild.cmds. Meanwhile.pl is a Perl script which will execute a Bash script in the background, log all the output and send an e-mail message when it is done.

The real workhorse is KernelBuild.cmds, which can be executed as a stand-alone script, although normally you would use KernelBuild.sh. It unpacks the source code tree, uses KernelConfig.exp to configure it, compiles the uniprocessor version of the kernel, packs it into a binary distribution file, packs the header files into a header-file-distribution, then repeats the process for the SMP version.

KernelConfig.exp determines how a kernel is to be configured, but KernelBuild.cmds determines how it is to be built and installed. The boundaries between the two are a bit blurred because of the way in which SMP support has changed from the 2.0.*x* series to the 2.2.*x* series, as mentioned earlier. If you wish to customise the build, it is these two scripts that you will want to change.

KernelBuild.cmds makes use of the fact that anything stored in a file called .name at the top level of the kernel source will be incorporated into the kernel name and can be retrieved later using the commands **uname -v** or **cat /proc/version**. I use this to record the kernel version string, including the distinction between uniprocessor and SMP versions. For the 2.2.*x* series, the Makefile

includes this distinction, but for the 2.0.*x* series it does not. KernelBuild.cmds uses a bit of **sed**, smoke and mirrors to smooth out the differences.

Finally, the binary and header file distributions are stored in the /dist directory, packed using tar, and compressed with bzip2. The binary distribution contains the kernel image, the System.map and all modules. It also contains a copy of KernelConfig.exp, so in the likely event that this script is updated, you will still have access to the exact version used to compile any particular distribution of the kernel. For the same reason, the log file of the configuration is also packed in the distribution. When the distribution is installed, these will find their way into the directory /log/*kernel-version*.

Installing the Kernel Binary and Header File Distributions

The kernels can be installed using the InstallKernel.pl script. InstallKernel.pl takes the full name of the kernel distribution file as input, with the ".tar.bz2" extensions. First, it checks that the distribution will not overwrite any existing file—if so, it aborts execution unless you specifically tell it to go ahead. It installs the kernel and its modules, and adds an entry to /etc/lilo.conf for this kernel. It is quite careful about how it does this. It creates a backup copy of /etc/lilo.conf, then scans it line by line until it finds a **root=** entry. It uses this to set the root for the new kernel. If it finds a later **root=** entry that specifies a different root partition, it will warn you, but will continue, using the first one it found. It will not add an entry if it finds an existing entry for this kernel image. The last thing it does is show you the differences between the saved lilo.conf and the one it has just created. InstallKernel.pl will not run LILO for you—you must do that yourself.

Another script, InstallHeaders.pl, will take care of installing the header files for you. The headers are installed as subdirectories of /usr/src/linux-headers. If you set the link /usr/src/linux to point to one of these installed sets of header files, you can compile your driver or program for a version of the kernel different from the one you are actually running. I make use of this to compile the ARLA AFS clone for all the kernels I support, without rebooting my machine.

Post-Installation Steps

Whichever distribution of Linux you are using, you will probably have to modify the way it decides which set of kernel modules to use. The details vary from distribution to distribution, so it is not possible to describe all the necessary changes here.

Since these kernels rely heavily on the use of modules, you may also need to create an initial RAM disk for your specific machine. This is certainly true if you

have a SCSI-based system. See the man page for the **mkinitrd** command for details.

Cloning the Kernel Repository

In order to clone the repository to build your own kernels, copy the contents of the /bin and /source directories, and modify them as you wish. KernelBuild.sh will need modifying in order to set the **MY_*** variables correctly. KernelConfig.exp may also need modifying to enable or disable any specific options—this may not be a trivial task. KernelBuild.cmds will need to be modified if you wish to actually change the way the kernels are built. The other scripts should never need to be altered.

Summary

At present, about 30 kernel source distributions are included in the repository, representing kernels from 2.0.34 to 2.0.36 and 2.2.0 to 2.2.7 with various patches. As the person who manages the machines running these different kernels, I find that this standardization has simplified my tasks considerably.

Tony Wildish received a Ph.D. in High Energy Particle Physics from Imperial College, London, in 1989. His career evolved from programming in Fortran to C and C++ while working at CERN. He became a systems administrator four years ago, and discovered Linux as a means of practicing his job while at home. Currently, he works at CERN for one of their experiments in preparation for the Large Hadron Collider, due to be commissioned in 2005. He enjoys Greek wine, Greek beaches and Greek food, as well as reading, and is especially fond of Terry Pratchets' Discworld series. His goal in life is to go on holiday and stay there. Tony can be reached via e-mail at tony.wildish@cern.ch.

Archive Index Issue Table of Contents

Advanced search

Focus on Software

David A. Bandel

Issue #65, September 1999

atsar, ext2resize, ippl and more.

Things continue to develop at a rapid pace. I took a look back at some of the packages I reviewed several months ago, and noticed many have had significant improvements. One package, the Ministry of Truth, a job tracking system, has morphed into a simple way to create databases and tables and make use of them. The original job tracking database is included and can be expanded. Other packages haven't made such a drastic metamorphosis, but have improved. So if you found them lacking back then, take another look.

atsar:ftp://ftp.atcomputing.nl/pub/tools/linux/

For those Linux users who have administered SCO boxes or other systems that have a utility known as the System Activity Report (SAR), you know what you're missing. For those who haven't, SAR is a great tool to track your system's health. A clone or port is long overdue. The way to get the best indication of your system's health is by putting an entry in crontab to run **atsar** every twenty minutes. This program will list how resources are being used, so you can prioritize what to buy to fix any bottlenecks found in the system. The atsar report lists percentages for user, system, nice and idle. Armed with this information, you can more easily justify system expenditures. It requires glibc.

ext2resize:http://www.dsv.nl/~buytenh/ext2resize/

Still in development, **ext2resize** shows a lot of promise. This utility purports to allow you to resize (shrink or extend) an existing EXT2 partition. Being the coward that I am and not having a disk or partition handy with expendable data, I tested only against a file I created, carefully following the instructions with the package. I would need to throw in a disk to test it properly. Problem is, I don't have one. It requires glibc.

ippl:http://www.via.ecp.fr/~hugo/ippl/

ippl is an IP protocol logger designed as a replacement for IP Logger, which logs IP packets. In contrast to **iplogger**, it is highly configurable. It will log any or all TCP, UDP and ICMP messages using the syslog facility, depending on how you have configured it. With this available, all you need is a Perl script or two to search (using **grep**) through the messages and check for anomalous behavior. Crackers often use scripts that essentially outsmart programs looking for sequential port scans by doing slow scans over a period of hours or days. With some as-yet-unwritten scripts, ippl could detect these scans. Now, if I just knew Perl. It requires glibc and libpthread.

galway:http://erin.netpedia.net/

galway is a small program that is a step toward becoming a usable web-page creation tool. It uses pull-down menus to help you create pages. These pull-down menus aren't complete as of this writing, but should be soon. If you don't like what you see, it is easily changed. You can add pull-downs, remove them, change them, or add/delete/change items on the pull-down menus. Since on any given set of web pages the top and bottom portions of the page (those containing the headers and footers) change rarely or very little, this program allows an upper and lower template to be added (respectively) above and below the body of the page. It requires guile, guile-gtk and gtk.

gperiodic:http://www.bgw.org/projects/gperiodic/

Got a student just starting out in chemistry who needs a copy of the periodic table of elements? **gperiodic** doesn't have it all (valences are missing, for one thing), but it does give the correct name, atomic number, weight, and boiling and melting points. This quick reference may be all your budding DuPont needs for a while. It requires libgtk, libgdk, libglib, libgmodule, libdl, libXext, libX11, libstdc++, libm and glibc.

syswatch:http://www.weirdo.net/scripts/

syswatch is a nice utility that displays in a web browser window what is going on on your system. System information shown includes uptime, kernel version, RAM and swap. A second section shows file system information. A third section displays the output from **w** command. A final section displays resource hogs based on CPU usage, memory usage, and amount of time on the processor. The only change I would make is to add an HTML tag to update the page automatically every minute or so. It requires a web server that allows CGI scripts (Perl) and a web browser.

xps:http://www.netwinder.org/~rocky/xps-home/

xps displays a process tree in an easy-to-read format. Unlike other process tree viewers/display programs, this one makes good use of diagonal lines to keep things grouped and on the screen without much scrolling around. Color is used to denote various states (running, sleeping, etc.). Double-clicking on a process pops up a window with specifics on that process and options which allow you to change priorities (**renice**) or send signals to the process, if you have sufficient permissions. It requires libXmu, libXm, libXext, libXt, libX11, libSM, libICE and glibc.

David A. Bandel (dbandel@ix.netcom.com) is a Computer Network Consultant specializing in Linux. When 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

LINUX JOURNAL

Dynamic Graphics

Reuven M. Lerner

Issue #65, September 1999

Generating graphics, charts and graphs for your web site is easy following Mr. Lerner's instructions.

Mark Andreessen, a co-founder of Netscape, is often credited with having turned the Web from an academic playground into a mass medium. But just what did Andreessen do? After all, Tim Berners-Lee invented the browser, HTML and URLs. You could even argue that the original browser was superior in some ways, in that it allowed people to write pages of HTML as well as read them.

Historians might take issue with this, but I would argue that Andreessen's greatest idea was allowing for graphics alongside text in web documents. As a text-only medium, the Web was interesting mainly to physicists and other academics, but with the introduction of graphics, it began to appeal to an entirely new segment of the population.

Today, graphics are not just used for decoration, but often stand on their own. Nearly every professional web site now hires one or more graphic artists to design the site—even when the site will deal mainly with text. Some sites would not be possible or even worthwhile were it not for the use of graphics. In some cases, these graphics are dynamically generated, produced by a program, instead of sitting in a static file on disk.

This month, we will look at ways in which we can create such dynamic graphics with CGI programs. We will look at the GD library, which allows us to create arbitrary images, and will quickly move on to creating different kinds of dynamically generated charts and graphs. After looking at some simple examples of such charts, we will examine a more sophisticated example, one which draws its inputs from a relational database.

Perl, Dynamic Graphics and GD

Writing a CGI program that outputs HTML is not particularly difficult, as we have demonstrated in many previous installments of “At the Forge”. Here, for example, is a simple program that, when invoked, returns some HTML to the user's browser:

```
#!/usr/bin/perl -wT
    use strict;
    use diagnostics;
    use CGI;
    use CGI::Carp qw(fatalsToBrowser);
    # Create an instance of CGI
    my $query = new CGI;
    # Send an appropriate MIME header
    print $query->header(-type =>
    "text/html");
    # Send some content
    print $query->start_html(-title =>
    "This is a test.");
    print "<H1>Testing!</H1>\n";
    print "<P>This is a test.</P>\n";
    print $query->end_html;
```

If we want to return graphics to the user's browser, we must modify the “Content-type” header in the HTTP response, generated with the call to “header”. If we want to generate a GIF, we will have to change our call to header such that it outputs “image/gif” instead. By the same token, we can tell the user's browser that a JPEG (image/jpeg) or PNG (image/png) graphic will be sent. Once we have described the content to the user's browser, we must generate a graphic of this type. How can we do that?

Perl's scalar variables can contain any data we might like. If we were more familiar with the GIF standard, we could stick a GIF into a scalar, then send that value to the user's browser. Of course, most of us are unfamiliar with the intimate details of the GIF standard, which makes this a less than ideal solution. A better idea would be to take advantage of Perl's object-oriented capabilities, using someone else's solution to the same problem.

Sure enough, Lincoln Stein (author of CGI.pm, the standard module for CGI programs) has written and distributed GD.pm. This module, available on CPAN (see “Resources”), gives us access to the popular “gd” libraries for C written by Thomas Boutell.

GD gives your program the ability to draw in a manner similar to popular drawing programs. You can choose from an array of paint brushes, colors and built-in shapes, as well as any fill shapes you have drawn. GD has its own internal drawing format, but as we will see, it supports the conversion of drawn images into GIF format.

A Simple Graphics Program

A simple program that uses GD, gd-intro.pl, is shown in Listing 1. If you install it in your CGI directory and invoke it from your browser, you should see a blue-filled green square.

Listing 1.

As you can see, our program manipulates two objects—an instance of CGI and an instance of GD. Each object handles its own affairs, keeping its nose out of the other object's business. **$query**, our instance of CGI, neither knows nor cares what sort of data we are receiving from the user or returning to his or her browser. By the same token, **$image**, our instance of GD, does not know that its output is going to be sent to a browser. Such compartmentalizing of tasks is one reason why objects make programming easier and software more maintainable.

When we create **$image**, we declare it to be of type **GD::Image** and to be 100 pixels wide by 100 tall. GD will not warn you if your image is cut off by the boundaries of this declared "canvas"; when I first started to play with GD, I was puzzled by the fact that no output appeared. I finally realized that my image was 100x100, but I was drawing a circle with a 400-pixel diameter. GD dutifully performed the task I requested, which meant that no picture appeared in the end.

After declaring **$image**, we allocate some colors, using GD's **colorAllocate** method. Each color is defined as red-green-blue (RGB), with each of those parameters varying between 0 and 255. I find it useful to declare color names within a hash, as with **%COLORS** in gd-intro.pl, but you may prefer to assign them to individual variables or to work with colorAllocate directly.

Next, we tell **$image** that it should create GIFs in "interlaced" mode. Interlacing means that instead of drawing every horizontal line of an image, the computer should first draw all the even lines, and then all the odd lines. You can see this in action on an ordinary television set. When TV standards were defined, televisions were unable to draw all the horizontal lines at once. Because of this, your television draws all the odd horizontal lines, followed by the even ones, followed by the odd ones again.

Making a GIF interlaced is not related to the speed of your computer or its ability to display images quickly; rather, it has to do with the speed of a user's connection. If the user has a slow connection, a GIF will load slowly. Making the graphic interlaced allows the user to see the graphic as it loads. Otherwise, the graphic will not be displayed until it is completely loaded, which might take a while.

We also set the “transparent” color, which is the color selected to melt into the background. By setting white as the transparent color, we indicate that anything drawn in white should actually be drawn in the color of the background. Since GD drawings have a white background by default, setting white as the transparent color means our graphic will appear to be floating in the user's browser, rather than set against a white background.

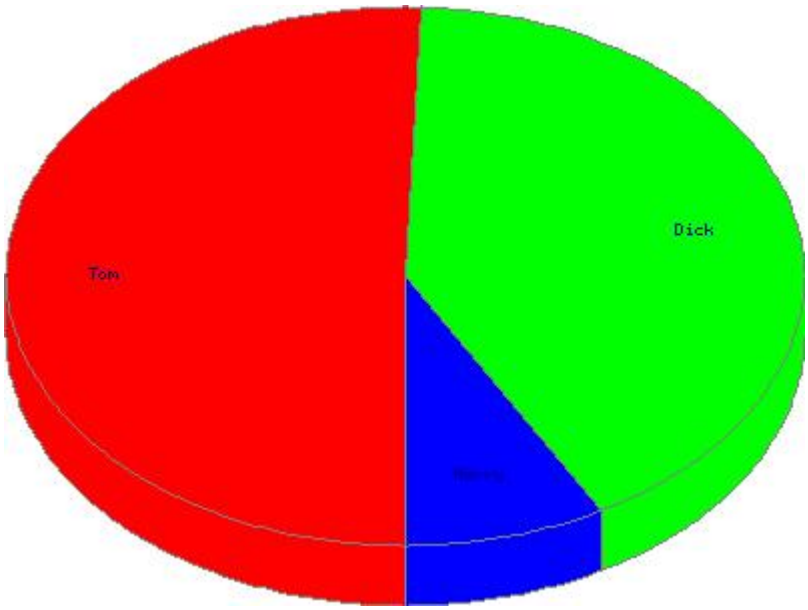
After all this, we can finally draw. We create a rectangle between 20,20 and 80,80, which should fill most of the 100x100 area defined when we created **$image**. We choose to draw the rectangle in green, using **%COLORS**, which we defined earlier. Finally, we fill the rectangle with blue by pointing GD to a point inside of the rectangle and asking it to fill the area.

GD has a number of other functions, including the ability to draw polygons, create custom brushes and fill with specified patterns. You can add text labels, which are incorporated into the final graph. You can even save graphics to disk in GD's own format, then load them again and continue to manipulate them before turning them into GIFs.

Charts and Graphs

GD is a wonderful tool for drawing on the Web. With it, you can create all sorts of marvelous things. Most of the web graphics I want to create are charts and graphs based on various types of data. I could use GD to create such graphs, but that would involve too much work.

Luckily, as is often the case with Perl, someone else had this problem and decided to solve it. Martien Verbruggen wrote and distributed the GIFgraph module, which allows us to create different types of charts based on a list of data points. GIFgraph uses GD, but provides us with an object-oriented interface to the new graph. This allows us to think in terms of graphs, styles and shapes—as opposed to GD, which would force us to think in terms of pixels and lines.



GIFgraph is actually a set of modules collected under the single “GIFgraph” name. One module handles bar graphs, another pie charts and so on, for nearly ten different types of graphs.

Listing 2.

In Listing 2, for example, we create a simple bar graph, with labels “a”, “b” and “c”, with respective values of 1, 2 and 3. We do this by creating an array, traditionally called **@data**. Each element of **@data** is an array reference, with the first element corresponding to the labels. Our program displays results from a single set of data:

```
my @data = (["a", "b", "c"], [1, 2, 3]);
```

We could easily compare two sets of data:

```
my @data = (["a","b","c"], [1,2,3], [4,5,6]);
```

GIFgraph is smart enough to use different colors for different sets of data. So given the above data, it will draw six bars—three each of two colors, with the values 1 and 4 associated with the “a” label, the values 2 and 5 associated with the “b” label, and the values 3 and 6 associated with the “c” label.

Before we can send the output to the user's browser, we must send a MIME type. Because it relies on GD, GIFgraph can produce output in GIF format. We tell the browser what to expect with the following command:

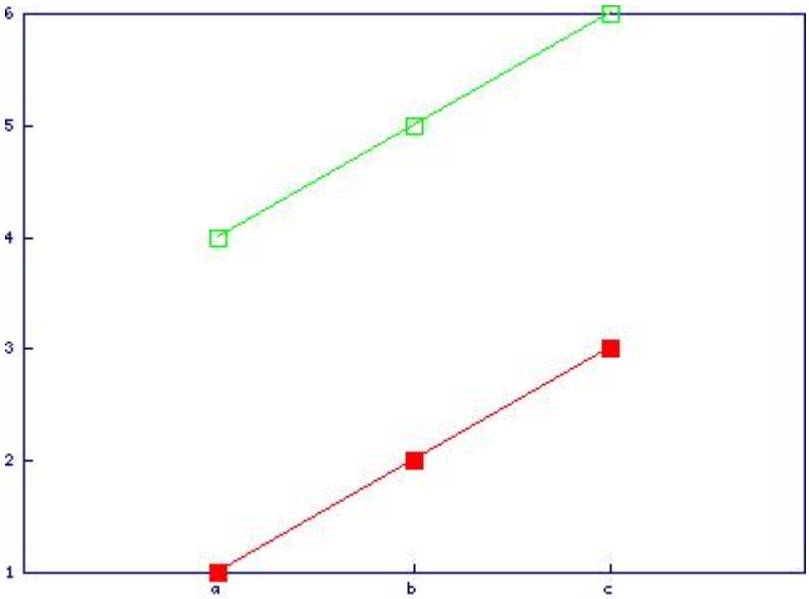
```
print $query->header(-type => "image/gif");
```

Now we will create our graph object and send its GIF output to the user's browser:

```
my $graph = new GIFgraph::bars;
print $graph->plot(\@data);
```

Notice how we pass an array reference to **@data** by prefacing it with a backslash (**\@data**). Passing **@data** as a reference ensures it will be handed to the **plot** method as intended.

In this example, we created a bar chart. What if we want a different kind of chart? We can do that by importing a different Perl module (e.g., **GIFgraph::lines** instead of **GIFgraph::bars**) and making **$graph** an instance of the new type.



Note that calling **$graph->plot** creates a graph based on **@data** but does not send it to the user's browser. This method returns the resulting GIF to its caller, allowing us to save it to disk, send it to the user's browser or manipulate the resulting GIF in Perl or external tools. Since the CGI standard mandates all output to STDOUT be sent to the user's browser, we can display the chart on the user's computer by printing the result from this call.

Charting Based on a File

Now that we have seen a simple program that produces a chart, let us look at a slightly more complicated example, one which mirrors some real-world situations. Assume we want to create a graph based on a text file. For example, assume we are implementing part of the reporting function for a web-based voting system. The results of a given election will be placed in a text file, called votes.txt:

```
Tom    123456
Dick   100000
Harry  20000
```

The election data is stored in the above file, with the candidate's name and the number of votes he received separated with one or more tab characters. This

allows the candidates' names to contain space characters, such as between first and last names.

Listing 3.

We could use a bar chart with this data, but it would not be nearly as useful as a pie chart, in which each candidate is given a proportional part of the pie. As you can see in Listing 3, our program vote.pl is not very difficult to create and produces results relatively quickly.

It does this by iterating through each line of votes.txt, using Perl's built-in "split" function to turn a scalar value (the line from votes.txt) into a list value. In this case, we split that line across tabs, putting everything before the tab in **$name** and everything after the tab in **$votes**. We then use the "push" function to add these values to **@names** and **@votes**, respectively, which are built up with every iteration through votes.txt. If there are four candidates in votes.txt, this loop is executed four times, and **@names** and **@votes** each has four elements.

When we exit from the loop, we create **@data** by inserting references to **@names** and **@votes**. As always, the first element of **@data** is an array reference containing the names. Subsequent elements of **@data** contain values; in this case, we have only one value, **@votes**. We create the graph by creating an instance of **GIFgraph::pie** and then plotting it to the user's browser.

Retrieving Data from a Database

The above example introduced us to the notion of creating a chart based on data stored on disk. While this is certainly the right idea, storing such data in a text file has its drawbacks. It is more common and more useful to put such data in a relational database.

Creating a chart based on a table in a relational database is not very different from creating one based on a text file. The main difference is with the loop we use to iterate over our input data. In vote.pl, we iterated over each line of votes.txt, turning each line of text into a name,value pair, which we then added to **@data**. When we retrieve information from a database, the information is already split into name,value pairs for us.

Before we can begin to write db-vote.pl (a database version of vote.pl), we must create a table in our database. As usual, I will use MySQL, a "mostly free" database described in Resources. MySQL's syntax is standard enough for most purposes, and most of the following should work with other databases as well.

Relational databases expect to receive input in SQL, the "structured query language". SQL is not a programming language—so while we can create all

sorts of queries to manipulate data in our table, we must embed those queries within a program written in a full programming language. Perl's DBI (“database interface”) module allows us to embed SQL statements inside our Perl programs.

We can create a new table by issuing the following SQL command:

```
CREATE TABLE Votes (
         candidate_name VARCHAR(30),
         votes_received BIGINT UNSIGNED
         );
```

While we could send the above to our database server from within a Perl program, it is more usual to type it directly from within an interactive database client. MySQL comes with an interactive client called **mysql** which allows you to send queries to the database (and receive responses) without having to embed your statements inside a Perl program.

After you issue the above SQL query, the database server will create a new table, Votes, with two columns. The first column, **candidate_name**, allows for up to 30 characters. The second column is defined to be a **BIGINT UNSIGNED**, that is, a large integer. We name this column **votes_received**.

We will now take a leap of faith and assume that, after the polls close on election night, our database table will magically be filled with appropriate values for each candidate. (In a real application, we would probably design things differently, storing each candidate's name in a second table and perhaps even storing each vote in its own row. We will ignore real-world concerns for the time being, so as to concentrate on how to create a graph with this data.)

Assuming our table has been populated with a list of candidates' names and their votes, how can we rewrite vote.pl so it takes its input from a database? As mentioned above, we will rely on DBI, Perl's database interface, which provides a uniform, object-oriented interface to most popular relational databases. Each database is described in a DBD, or database driver, and is imported automatically when we open a connection.

Opening a connection to the database creates a “database handle” object, traditionally called **$dbh**. We use this object to create a “statement handle”, traditionally called **$sth**, with which we send the SQL to the database server. Our query, in this case, is rather simple:

```
SELECT candidate_name, votes_received
    FROM Votes
```

When it executes this query, the database server will return a two-column table to the user—in this particular case, the entire contents of the Votes table. Each

row of the table corresponds to a line in the text file votes.txt which we saw earlier.

DBI provides us with a number of methods by which to retrieve data from **$sth**. The most commonly used methods retrieve a row as an array, either in its usual form (using **$sth->fetchrow_array**) or as a reference (using **$sth->fetchrow_arrayref**). While the **arrayref** method is more efficient, beginning Perl programmers often prefer to avoid references, which sometimes confuse them. In both cases, the order of elements in the returned list is determined by the order in which columns were named in the query.

Listing 4.

The rest of db-vote.pl (see Listing 4) continues in almost the same way as vote.pl, pushing the values in each row onto **@names** and **@values**, then using those to create **@data**.

It is generally preferable to put such information in a database, because of the reliability and flexibility offered by relational databases. Remember, though, there is no free lunch: a relational database is inherently much slower than a flat ASCII text file. Moreover, our CGI program opens a connection to the database each time it is invoked, an expensive and time-consuming operation. For these reasons, vote.pl will almost certainly execute faster than db-vote.pl. Whether this is an appropriate trade-off depends on the number of visitors to your site, as well as the nature of your web applications.

Modifying the Graph

Now that you have seen how to create simple graphs based on various inputs, let us spend a few moments discussing how you can modify the outputs. GIFgraph allows you to change just about every aspect of the graph, including the colors, placement and style of the legend, and the way in which the axes are marked. This is done with the **set** method. Of course, certain settings are active for only certain types of graphs; for instance, there are no axes on a pie chart, meaning that setting the axis labels will be meaningless.

Here is one example invocation of set:

```
$graph->set(x_label => "Candidates",
        y_label => "Number of votes",
        title => "Voting results",
      logo => "corplogo.gif",
      zero_axis => 1);
```

The GIFgraph manual page, available by typing **perldoc GIFgraph** after installing the package, describes these and many other options in detail. However, the above is probably a good starting point and demonstrates how the various

factors describing a chart can be set. In the above example, we label the X axis **Candidates**, the Y axis **Number of votes**, include our corporate logo on the chart, and ensure the axes will always begin at the origin (0, 0). There are also options to choose colors and fonts, as well as define how often ticks should appear on each axis—if you read the manual, you will likely be overwhelmed by the wealth of options.

Conclusion

As you can see, it is not particularly difficult to create graphics on the fly from within our CGI program. Even more impressive—as well as generally useful—is the ability to create many types of charts and graphs in only a few lines of code.

Next month, we will take a further look at dynamically generated graphics, looking at a simple application that tracks a user's stock portfolio. That application will revisit two topics we discussed last month, namely HTTP cookies and saving state to a database.

Reuven M. Lerner 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 later this year. Reuven can be reached at reuven@lerner.co.il. 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

Voice-Over IP for Linux

Greg Herlein

Ed Okerson

Issue #65, September 1999

Make your long-distance calls over the Internet using this new technology for Linux.

We've all got friends, family and Linux collaborators scattered around the world, and we all like to talk. We love to pick up the phone and gab—but the costs for long-distance phone calls are high, and international rates are worse. Many, if not all, of the people we would like to call have Internet access. We can now use our normal phone over the Internet to make phone calls using Linux.

Voice-over IP is the answer, but it is a technology that has been slow to migrate to Linux. Sure, applications are available that can already do this using a sound card, but frankly, the headset and sound card solution is a kludge. Headsets don't ring when an incoming call comes in, and most sound cards do half-duplex voice that, at best, is annoying and cards don't connect to your personal or business phone system. Other technologies have been available for Win32 platforms, but not for Linux. That is changing quickly, especially with the recent pre-alpha release of new Linux drivers for the Internet PhoneJACK and Internet LineJACK voice-interface cards from Quicknet Technologies, Inc.

The Hardware

Quicknet's Internet PhoneJACK card provides a low-cost, full-duplex audio interface and telephone-line interface to a normal phone. The card has a POTS (plain old telephone service) interface (RJ-11), where you plug in a normal analog telephone and use it to make phone calls over the Internet. When calls come in, the phone rings. When you want to make an outgoing call, you can dial the digits from the phone or from software control. In all respects, it is a phone—it just works over the Net. If you are calling a party that also has an Internet PhoneJACK/LineJACK, or compatible software, then your call is free! If you want

to call someone on their normal phone line (not over their computer), you can do that too, via a PC-to-Phone service like Net2Phone. In that case, you would have already set up an account with the provider (via a browser), your call would go over the Net to them and they would get the call to your party over their system, charging you a low rate for the service. Obviously, the best case is if both parties have Quicknet cards, because then you avoid the costs of the PC-to-Phone service provider and get the highest quality. Another way to get it free is by using a Quicknet Internet LineJACK card.

A huge advantage of Internet PhoneJACK/LineJACK cards is the audio-compression capabilities (CODECs) built into the cards. These include G.711 (64Kbps), G.723.1 (6.3Kbps and 5.3Kbps) and TrueSpeech (8.5/4.8/4.1Kbps) audio compression in hardware. These compression technologies may be used with no royalties or fees whatsoever—the license cost is part of the hardware cost. This is a big deal, because it allows small developers to build compressed-speech applications without having to spend thousands of dollars on licensing the CODECs. In other words, it makes it free to use the same technological advantages as the big boys.

The Internet LineJACK card adds an extra twist: it has a PSTN (public switched telephone network) interface. The PSTN plug (RJ-11) is connected to your normal phone line, allowing you to place and receive normal phone calls using the card. All DTMF and tone-generation capabilities are built into the Internet LineJACK. It is also designed for software-controlled compatibility with different phone networks around the world. (Different countries have different electrical and tone signals with which the Internet LineJACK is capable of interfacing by selecting the appropriate parameter set.) With the appropriate software, you can use the Internet LineJACK to create a one-line gateway between the Internet and the normal phone system. To solve the problem of not being able to make PC-to-phone calls without an applicable service provider, all you need to do is set up an Internet LineJACK card on a system in the area you want to call. Your voice call “hops on” the Internet at your computer, then “hops off” at the remote Internet LineJACK and out onto the local PSTN in that community to complete the call—all for free.

It is important to realize that a single G.723.1 compressed audio “call” uses only 6.4Kbps. Even on a normal 33.6Kbps modem call, there is sufficient bandwidth for multiple simultaneous calls over a single link.

Until recently, drivers for these cards were available only for Windows 95/98/NT. However, Quicknet has recently released some early pre-alpha versions of their Linux drivers, along with a sample application to provide an example of how it all works.

Getting the Driver

You can get the Linux drivers at www.quicknet.net/develop.htm. Be warned: these are still early, pre-alpha releases and are not for the faint of heart. At this early stage, these are recommended only for people who wish to play with new technology and are not afraid to fiddle around with their system. Of course, that is most Linux people, but we had to warn you. The driver is distributed in a compressed **tar** file that includes sample application source code and a simple HOWTO.

Licensing

Many Linux folks may criticize Quicknet because their device driver is not Open Source. Quicknet has made a carefully reasoned business decision to keep their device driver binary-only and not reveal the source code. A full API will be published to detail how to use the device driver, but the source code will be closed. To their credit, Quicknet realizes the tremendous opportunities with Linux and has made the investments needed to get the Linux drivers written. Quicknet has also continued to invest in this area by hiring Linux developers to work in-house.

However, the sample code accompanying the device driver is released under the Lesser GNU Public License (LGPL). While there may be some who demand the drivers be released Open Source, Quicknet believes strongly that having stable, reliable binary-only drivers is good enough for most users, and Quicknet is committed to supporting such a driver.

Requirements

The device driver for the cards requires a 2.2-series kernel, because some of the low-level features take advantage of changes in the new kernel. The latest modules are compiled with MODVERSIONS support, so assuming your kernel is, the kernel will load the module if the symbol tables are compatible between the module and the kernel. At the time of this writing, we are using 2.2.10 kernels for development.

Our modules have a dedicated “major number” so they can be used with the normal module tools such as insmod and modprobe. All you have to do is add a few lines to your conf.modules file, and the normal modules tools work with our device driver. See below for more information on how to perform the initial configuration.

You will need at least one Internet PhoneJACK/LineJACK card on each end of the connection. Note that the initial release of the Linux driver supports only the Internet PhoneJACK card, though by the time you read this, the Internet

LineJACK will be supported as well. These cards are ISA bus devices that use Plug-n-Play for configuration and use no IRQs. The driver will support up to 16 cards in any one system, of any mix between the two types.

Since the Quicknet cards are Plug-n-Play devices, you will need the **isapnp** tools package to configure the cards. This package probably came with your Linux distribution. Documentation is available on-line at metalab.unc.edu/LDP/HOWTO/Plug-and-Play-HOWTO.html.

Configuration

The Internet PhoneJACK has only one configuration register that requires 16 I/O ports. The Internet LineJACK card has two configuration registers. Isapnp reports that I/O 0 requires 16 I/O ports and I/O 1 requires 8. The Quicknet driver assumes these registers are configured to be contiguous, i.e., if I/O 0 is set to 0x340, then I/O 1 should be set to 0x350.

If you are new to the isapnp tools, you can jump-start yourself by doing the following:

- Run pnpdump to get a blank isapnp.conf file

```
pnpdump > /etc/isapnp.conf
```

- Edit the /etc/isapnp.conf file to set the register I/O addresses.
- If you have multiple Quicknet cards, make sure you do not have any overlaps. Be especially careful if you are mixing Internet PhoneJACK and Internet LineJACK cards in the same system.

Use of the Driver

To install and load the driver, perform the following:

- Unpack the distribution file using tar.
- Run the included **ixj_dev_create** script to create the device files in the /dev directory. This script will create /dev/ixj0 through ixj16.
- Run the isapnp configuration utility to configure the cards properly.
- Edit the /etc/conf.modules file to add a line that specifies the I/O (input/output) port(s) for the cards you just configured with isapnp, and a line to map the device number to the name of our device. You will need to add the following two lines:

```
options ixj io=0x300 ixjdebug=a0
alias char-major-159 ixj
```

This example assumes you have one card at I/O address 0x300; you will need to modify that value if you assigned card(s) to different ports.

- Load the module with insmod or modprobe, as you would for any other module.
- Verify the module loaded by running **lsmod**.
- Execute an application that uses the module (**tpjack** or **tpjackd**).

Example Code

Sample applications are included with the driver to demonstrate its use. The following are some of the immediately useful ones:

intercom.c: demonstrates the driver's capability to pass audio between multiple cards without passing the audio data through user space. The application only has to indicate which cards will talk to each other—the driver does the rest.

inter2.c: the same concept as intercom.c, only it passes the data through user space. In this example, the application has to deal with reading and writing to the device files to pass data between cards.

tpjackd.c: this is the server side of a very basic IP Telephony application. It simply waits on a TCP port for an incoming connection. When received, the phone rings. When the phone is lifted, it starts passing UDP packets with audio data to the tpjack.c program.

tpjack.c: this is the client side, corresponding to tpjackd.c.

Test Applications

tpjackd is the daemon application which listens for an incoming call. To use it, type this:

```
tpjackd dev port
```

dev is the name of the device and is usually /dev/ixj0, although if multiple cards are in the system, it might be /dev/ixj1, etc. *port* is the name of the port on which the daemon will listen for an incoming ring.

The daemon application now runs as a true daemon. It disconnects from the controlling terminal upon startup and runs only in the background, logging messages to syslog. Typical use during development is watching the logging in an xterm window by using the command:

```
tail -f /var/log/messages
```

tpjack is the calling application and is used by typing the following:

```
tpjack dev host port
```

dev is the name of the device and is usually /dev/ixj0, although if multiple cards are in the system, it might be /dev/ixj1, etc. *host* is the name of the host running the daemon (name, not IP address). Note that the names of both hosts need to be resolvable, either by DNS or the local host's files. *port* is the port at which the daemon will listen for an incoming ring.

These sample programs provide an example of how to use the driver, and have the added advantage of actually working well over the Internet. The authors have used it to converse between San Francisco, California and (roughly) Dallas, Texas. The voice quality was not as good as expected using real-time protocols (RTP), but it was certainly good enough to have an intelligible conversation.

Known Limitations

Work is progressing rapidly on the driver, so check the web site often for new versions. By the time you read this, we should be in beta testing with full support for all the card's features.

The sample code is crude and does not follow any of the standards for Voice-over IP (H.323, SIP, etc.). Support for such protocols will come later, though probably not in source code form, due to other licensing restrictions. The purpose of the initial sample code is to provide a simple means of exercising the drivers and doing simple voice.

Currently, no software for Linux supports the use of any PC-to-Phone gateway service providers (such as Net2Phone). Of course, that will also change soon.

The Future

The sky is the limit for what can be done with these cards. With the availability of Linux drivers, we can craft all manner of servers to perform telephony functions—over the Internet and with or without the normal phone network. Some of the things we might soon see include VoIP PBXs, voice-mail services, and PC-to-PC and PC-to-Phone gateways. However, the most exciting applications for this technology have probably not even been imagined yet. This is a wide-open area that is begging for the Linux crowd to start putting out cool new applications. Quicknet Technologies wants to put the device drivers in place, along with some simple libraries, to facilitate this innovation.

More Information

Additional information is available on the Quicknet web site at http://www.quicknet.net/. A new mailing list has been started to provide developers with a forum for discussing the development and use of the device driver. To subscribe, send e-mail to majordomo@linux.quicknet.net; in the body of the message, type

```
subscribe linux -sdk your_email_address
```

After verifying your subscription, you can send mail to linux-sdk@linux.quicknet.net. If you have any problems with it, please send e-mail to linux@quicknet.net, and we will help you however we can.

Internet PhoneJACK and Internet LineJACK are registered trademarks of Quicknet Technologies, Inc.

Greg Herlein (gherlein@quicknet.net) co-authored the sample applications above and wrote the documentation for the release. He is a long-time Linux programmer who has crafted Linux solutions on the high seas, remote mountaintops and in corporate offices. He just recently joined Quicknet Technologies, Inc. as a Member of the Technical Staff, developing Voice-over IP solutions (especially for Linux).

Ed Okerson (eokerson@quicknet.net) is the author of the Quicknet device drivers and is also a long-time Linux guru. He's built out and run an ISP in Texas and continues to build innovative Linux-based networking, voice and video solutions. He recently joined Quicknet Technologies, Inc. as a Member of the Technical Staff.

Archive Index Issue Table of Contents

Advanced search

cron: Job Scheduler

Michael S. Keller

Issue #65, September 1999

Have you ever wandered near your Linux box in the middle of the night, only to discover the hard disk working furiously? If you have, or just want a way for some task to occur at regular intervals, cron is the answer.

The cron daemon, **crond**, packaged with most Linux distributions, controls scheduling of regularly occurring jobs. When started upon entry into multi-user mode, crond scans the directories /var/spool/cron/crontabs and /etc/cron.d and the file /etc/crontab for work to do. **crond** then awakens every minute, performs the work its record of jobs says it should do at that time, mails the output (by default) to the owning user, then sleeps until the beginning of the next minute.

The implementation of crond packaged with Debian 2.0, the distribution I used when writing this article, carries the name Vixie Cron, after Paul Vixie, its author. I will use “cron” to refer, variously, to both the crond process and the cron facility.

History of cron

cron evolved to enable the execution of jobs at regular intervals. Have you had occasion to use the log files in /var/log? Most Linux distributions come with a ready set of cron jobs to tame those log files. Without cron jobs, the file system holding /var would eventually fill completely with log files. The potential uses for cron exceed the small customizations I have made to my home environment. If you want to automate something that runs more than once, turn to cron.

How to Use cron

Individual users may use cron to automate tasks. Normally, all users may make use of cron. If superuser has created /etc/cron.allow or /etc/cron.deny, then

access to the cron facility depends on the contents of those files. If /etc/cron.allow exists, your user name must appear in it for you to use cron. If /etc/cron.deny exists but /etc/cron.allow does not, your user name must not appear in /etc/cron.deny, or cron will not work for you. To edit your cron settings, use the **crontab** command:

```
crontab -e
```

This will create a cron table, or “crontab file”, which cron will read to find work. The crontab command looks first for the **VISUAL** environment variable, then for the **EDITOR** environment variable. It will use the editor named in those variables to provide editing of crontab files. Without one of these environment variables set, Debian 2.0's crontab uses the **ae** editor. Other distributions may have a different default behavior for crontab. Make the changes you desire, save the file and exit the editor.

Why do we not edit the crontab file directly? The reason is **cron** requires a specific format for its job entries. The crontab command performs syntax checking before allowing a newly edited crontab file to enter circulation. If the new crontab has a syntax error, crontab complains and asks if you want to edit again. To protect the crontab files, the crontab command makes root the owner of the crontab files.

Listing 1.

To view your newly edited crontab file, use this command:

```
crontab -l
```

The output should look something like Listing 1. Each crontab entry provides either an environment variable or a time-specific cron command. **cron** sets a few environment variables automatically. Others, such as **MAILTO**, can be set by the user. Normally, cron mails the output of each cron job to its owner. If you put the line

```
MAILTO="fred"
```

in your crontab file, the output of your cron jobs would go to user fred instead. More likely, you would want to suppress cron output. If you set MAILTO to null,

```
MAILTO=""
```

then cron will discard the job output.

The fields in a time-specific cron command appear in this order: minute (0-59 allowed), hour (0-23 allowed), day of month (1-31 allowed), month (1-12 or names allowed), day of week (0-7 or names allowed, with both 0 and 7

representing Sunday), and the command to run. The numerical fields also allow ranges of numbers, wild cards, lists and methods for running cron jobs at every *N*th interval, such as every third hour. The asterisk character works as a wild card, representing every occurrence of the field's value. For details, see the crontab(5) manual page.

The example below will run the **ls** command every minute of the noon hour on the first day of the month, discarding the output:

```
MAILTO=""
# Minute Hour Day-Of-Month Month Day-Of-Week
# Command
* 12 1 * * /usr/bin/ls
```

This next example will run the **free** command every other hour and mail the output to fred:

```
MAILTO="fred"
* */2 * * * /usr/bin/free
```

The system-wide crontab, stored in the file /etc/crontab, provides a slightly different method for running cron jobs. It does not have a special editor, so you must take extra care when editing it. In addition, it provides a user name field between the Day-of-Week and Command fields, to run jobs under a user ID other than root, without having to create a separate crontab file for that user. Edit it with your favorite editor and save the changes; cron will automatically update its job list.

Pre-Configured cron Jobs

The Debian and Red Hat distributions come with several pre-configured cron jobs to help control disk usage. Other distributions may provide similar help. The Debian **dh_installcron** command will install these jobs. Normally, you should not need to run this command—installing the cron package will take care of it for you.

These jobs, located in the file /etc/crontab, use the **run-parts** command to call all the scripts in directories /etc/cron.daily, /etc/cron.weekly and /etc/cron.monthly. For the most part, these scripts control disk usage, compressing and pruning log files in /var/log and cleaning up after indices from the man(1) command. The package maintainers who created these jobs configured them to run during the night, normally a slow time for other system activity. Some of the scripts generate a lot of disk activity, which can slow other I/O-intensive jobs. If you want them to run at other times, edit /etc/crontab or move the scripts among the directories that contain them.

Additional Reading

See the list of references in the sidebar for additional reading on cron. Additionally, the man page for the **at** facility may prove useful. **at** provides a one-time job-scheduling facility. If you do not keep your Linux system running 24 hours per day, you may want to review Anacron, which does not depend on specific time events to get its work done.

Conclusion

I have provided a brief introduction to the cron facility, a typical part of Linux and other UNIX operating systems. It will provide a starting point for time-related work you want your Linux system to perform. In brief, if you want to schedule repetitive tasks so as not to type the same commands again and again, use cron.

Resources

Michael S. Keller works as a technical analyst with Sprint Paranet, a wholly owned subsidiary of Sprint, a nationwide network services provider based in Houston. He has used UNIX variants for nearly nine years and enjoys communing with cats, motorcycles and the universe. You may reach him at mskeller@sprintparanet.com.

Archive Index Issue Table of Contents

Advanced search

Red Hat Linux 6.0

Jason Kroll

Issue #65, September 1999

In keeping with the high standards of modern distributions, Red Hat Linux 6.0 is relatively easy to install, preconfigured, aesthetic and functional.

- Manufacturer: Red Hat Software, Inc.
- E-mail: sales@redhat.com
- URL: http://www.redhat.com/
- Price: $79.95
- Reviewer: Jason Kroll

Red Hat Linux 6.0 is Red Hat's latest distribution, and it has improved noticeably since the days of version 5. In keeping with the high standards of modern distributions, Red Hat Linux 6.0 is relatively easy to install, preconfigured, aesthetic and functional. It comes with the standard Linux applications (including Netscape) and also includes a special applications CD, with over 50 various commercial applications (most of which are demo versions which expire or are disabled). GNOME (running by default with the Enlightenment window manager), Red Hat's desktop environment, is well-configured and attractive. In addition to aesthetic improvements, Red Hat made some significant technical changes in its newest distribution, including

the complete adoption of the EGCS (Experimental GNU Compiler System). Still, many things will be familiar to users of previous Red Hat releases.

Installation

Red Hat Linux 6.0 comes with three CDs (and a floppy, just in case), the first being an auto-booting installer which looks rather like previous Red Hat installers. It presents three installation options: Workstation-class, Server-class and Custom. The 400+ page installation guide contains very little information regarding the installation options, but having tested them, I *really* recommend a custom install.

Before proceeding with any kind of installation, however, the installer offers a choice of eleven languages which are mostly implemented (though some need more work than others). These are mainly just for the installation process, although GNOME does support a few languages; so if you install in another language for the fun of it, your system could end up running in it. After language selection, the installer offers a choice between installation and upgrade.

Figure 1. **Screen Shot**

Upgrading is a rather quick process; a large part of the convenience aspect of Red Hat is the ability to upgrade, whether one is dealing with individual packages (via RPM/GnoRPM) or a complete system. Upgrading is rather automated compared to installation, which presents various options.

The workstation-class installation is quite functional and easy to use but is not as complete or fun as a custom installation; KDE, among other things, is noticeably missing. The server-class installation is meant for servers and will be of little interest except to network enthusiasts. It is not exactly up-to-date, and still runs FVWM2 (AnotherLevel) instead of the newer desktop environments; it is reminiscent of Red Hat releases from quite a while ago. The custom installation is probably what most users would want, and is simple enough to make if one knows what hardware is inside the machine.

Custom installation allows the user to choose packages either categorically or one by one; in the latter case, the installer keeps track of dependencies between packages. The user is also given the choice of which programs to be launched automatically at startup, and is required to partition his own drives. A choice of either Disk Druid or **fdisk** is given for partitioning the drives. Disk Druid is menu-driven and simple enough, as long as one is familiar with partitioning; the time-tested fdisk is just as adequate. Although both partitioning programs perform the same task in basically the same way

(selecting partitions, sizes and mount points), using a menuing system seems to be easier for most people.

Video configuration can be a bit problematic. Curiously, although the lists of available monitors and video cards are quite long, the installer cannot probe for the video card. Probing for a card instead of querying the user should not be very difficult to implement, since X can probe successfully. The video mode tests failed even though my card and monitor were listed, but after installation, X worked fine. Also, if one could test various preconfigured monitor frequencies against the standard X test pattern, a better picture could be had. Since you will presumably have this Linux system for a while before reinstalling, it would be worth the effort to have an optimally configured display. Too bad this option is not available.

If you do not know your hardware, installation can be a hang-up. The only thing the installer could successfully detect was my mouse; everything else had to be entered manually. Again, this is fine if you know your hardware, and it is probably even safer than probing. However, new Linux users to whom Red Hat is often recommended and people who do not know what is inside their computers might prefer the computer to figure out for itself what hardware is present. Certain other installers probe successfully, so accurate probing is possible. Since ease of use has long been one of Red Hat's main attractions, it seems the installer could stand to be brought up to a level on a par with the overall quality of the distribution.

Despite the need to enter hardware information manually, installation is not exactly difficult and an experienced user could reasonably expect to complete an installation in approximately thirty minutes. Once the installation process is at an end, the option is presented to have X start up by default at boot time. If you answer yes, reboot and log in, you will be greeted by a mysterious footprint on the desktop, shaped oddly enough like a G with toes.

GNOME

Red Hat has taken an active role in supporting the production of GNOME, a high-quality desktop environment based entirely on free software. Red Hat Linux 6.0 installs GNOME on workstation and custom installations. GNOME is not actually a window manager in itself—it is a desktop environment which allows you to use the compatible window manager of your choice (Enlightenment, by default).

One of the particularly nice characteristics of modern distributions is that their desktop environments come thoroughly configured. This means you can bring up menus, point and click, drag and drop, etc. without having to configure the menus, and the menu options actually correspond to the programs on your

system. On Red Hat Linux 6.0, GNOME is better configured than KDE or AnotherLevel (FVWM2), both of which menus are incorporated into GNOME's menuing system. GNOME also has the remarkable program GnoRPM, which is a Red Hat Package Manager for GNOME. This graphical program offers a simple point-and-click system for installing, upgrading, uninstalling, querying, verifying and searching through RPM packages. In conjunction with LinuxConf and various control panels, this makes for easy point-and-click administration and configuration.

Technical Changes

A large part of the improvement Red Hat made from 5.2 to 6.0 comes from the technical changes in the various programs which make up the distribution. The core of the improvement is the 2.2 kernel (2.2.5-15 to be exact), which supports more hardware and file systems and is even better at networking than previous Linux kernels. It also has better SMP support and countless other improvements in areas ranging from networking to frame buffers, and is even more modular and easier to reconfigure and recompile. Recompilation is less necessary due to the new modular approach. Developers will likely appreciate that Red Hat has moved completely to the Experimental GNU Compiler System (EGCS), which offers advanced platform optimizations, integrated FORTRAN and a significantly improved C++ compiler. These days, as innovative hardware solutions push past the limits of conventional desktop processor speed and storage space, SMP and RAID support are increasingly valuable and, thanks to the new Linux kernel, available. In conjunction with Glibc 2.1.1 and the latest stable versions of various libraries and programs at the time of its release, Red Hat Linux 6.0 is up to date.

Security is an ever-present problem with network computers, and in this distribution, root access via TELNET has been removed and the X screen automatically locks when the screen saver comes on. Also, passwords are shadowed and, optionally, use MD5 encryption (as opposed to DES). For convenience, console users do have access to peripherals and can reboot, although this can be changed. Security risks are always a hazard, so it is a good idea to check periodically for recent patches. In fact, a few small potential security problems shipped with 6.0; the fixes (via RPM) are on Red Hat's web site, along with a list of rather minor errata.

Application CD

Included in Red Hat Linux 6.0 and Red Hat EXTRA is an application CD with over 50 Linux applications, ranging from developmental software to productivity software, specialized commercial applications and more. According to the box, the CD is valued at over $1,000, and perhaps if it were full of commercial software instead of disabled and expiring demo versions, it would be worth

that much. Nevertheless, the disc is an amazing testament to the proliferation of applications available for Linux, and some software packages are mostly or even fully functional while some don't work at all. The CD should be looked at as a bonus, because you could just download most of the software over the Net, if you knew it existed in the first place. The inclusion of commercial demos with Linux distributions is a good idea, and software vendors might want to pursue other distributors as well.

Support and Manuals

In the past, the majority of complaints about Red Hat seem to have involved the issue of support; perhaps this implies that the distribution itself leaves little to complain about. This time, Red Hat stepped up the efforts to provide support to registered users of Official Red Hat Linux 6.0. This may partially explain the rather high price of $79.95 for a collection of mostly free software.

Included with the Red Hat package is a large bright yellow slip of paper stating, “For Installation Support Go To: http://support.redhat.com”, from which one might surmise the source for installation support. In order to receive support, one must first register via Red Hat's web site. The registration program seems to work, and once registered, a user is entitled to 30 days of installation support via telephone and 90 days of installation support by way of e-mail. I would certainly expect that after 90 days, someone would have his system installed. Actually, support goes a bit further than initial installation; Red Hat is willing to help in the configuration of printers, sound cards and other hardware such as floppy and CD-ROM drives. There was no mention of Ethernet help, though this is usually easier than dealing with sound cards, so I hope Red Hat intends to help with this too.

Installation will take much longer than thirty minutes if you take the time to read the 400+ page “Installation Guide” and the 300-page “Getting Started Guide”. The Installation Guide is a comprehensive walk-through of the installation process, with a significant space dealing with system configuration and administration. The Installation Guide is rather thorough and contains enough information to turn a neophyte into a competent administrator of his own system. The Getting Started Guide is smaller, simpler and a bit friendlier. It covers many aspects of Enlightenment and GNOME, X, shell usage, administration, configuration and the like. Red Hat calls it “easy-to-read”--that is a fair assessment, to say the least.

In the event that someone finds 30 days of phone support, 90 days of e-mail support and 700 pages of textual support inadequate, Red Hat offers various commercial support packages ranging in price from $2,995 to $60,000. Obviously, these cover more than basic installation.

Conclusion

Red Hat Linux 6.0 is a modern, up-to-date, flexible distribution which finds itself at home in a number of areas ranging from small servers to home desktops to the business world. Many businesses and institutions rely on Red Hat, as do countless home users. At the very least, it has recent versions of packages and puts libraries in the right places, so things work. It does have a commercial feel to it—you *know* when a machine is running Red Hat. Also, it does not take a minimalist approach, so it could be a bit more complicated than a home user might want—some might even find it a bit bulky. Actually, for GNOME/Enlightenment to function in a timely way, 32MB of RAM seems inadequate. However, the distribution on the whole is reliable and functional. The price is a bit painful, so one might want to consider the many alternatives. But, if you need to be sure of a functional system with phone and e-mail support, manuals, applications and a Red Hat bumper sticker, the price may be worth it.

Jason Kroll spends his weekdays in the product testing lab at *Linux Journal* where he is very happy to be working in support of the Linux movement. His evenings are supposedly spent finishing his economics studies (but not really, other classes are more fun, you see). When reviewing a distribution, he thinks it's very important to test all of the games, and wishes distributors would include more games with their distributions. He can be reached at info@linuxjournal.com.

Archive Index Issue Table of Contents

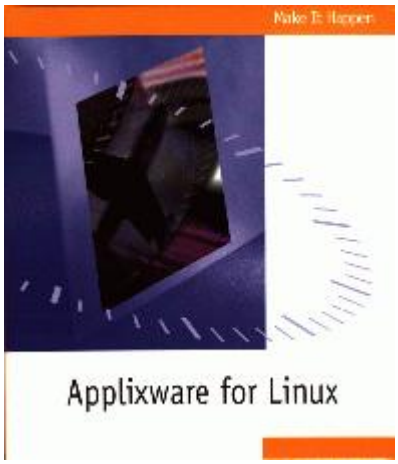
Advanced search

Applixware 4.4.1 for Linux

Dean M. Staff

Issue #65, September 1999

The package has been redesigned, and a WordPerfect-style template for the keyboard provides all the function and “accelerator” keys for quick reference.



- Manufacturer: Applix Inc.
- URL: http://www.applix.com/
- Price: $99.00 US
- Reviewer: Dean M. Staff

I came to the Linux world as a user—not a system administrator or a programmer. I was looking for a more stable alternative to Microsoft Windows. Once I found Linux, the search for productivity software soon began. I still needed to interact with others using the typical MS Windows applications, so importing and exporting to these formats was a priority. I took a serious look at Applixware and present here the results of my investigation.

Right out of the box, some changes from version 4.3 are apparent. The package has been redesigned, and a WordPerfect-style template for the keyboard provides all the function and “accelerator” keys for quick reference. Someone definitely put some thought into the preparation of the user's manual. The

usual glued spine binding has been replaced with a large spiral binding, allowing the user to either fold the cover over, leaving only the desired page face up, or lay it flat without the pages flipping over while attempting to use the computer. The biggest improvement is a single consolidated table of contents and index for the complete manual. (Version 4.3 had three distinct sections, each with its own table of contents and index, making it quite difficult to use.) While many positive improvements have been made, the manual is still lacking sections referring to Applix Data (the database module), Applix Mail (the built-in e-mail module) and Applix Builder (the graphical programming module).

Rating

I used the following rating system in regard to importing files to the different Applixware applications:

***** (Excellent): File was seamlessly imported, with no errors or anomalies encountered.**** (Good): File was imported, but some formatting was lost.*** (Fair): Import was successful, but all formatting was lost.** (Poor): Import completed with corrupted data that could be repaired.* (Fail): Import failed or data was corrupted beyond repair.

Importing to Applix Words ****

I was unable to import a Framemaker 5 file, but this may have to do with the fact that I used an application other than Framemaker 5 to create the source file. Importing an RTF file was flawless, and importing Microsoft Word 95 and Corel WordPerfect 6/7/8 files was nearly flawless, with only one glitch—it added a couple of tabs after the text on a right-justified paragraph.

Importing to Applix Spreadsheets ***

Importing spreadsheets from other applications was not as clean as in Applix Words. Microsoft Excel 4 and 95 files were imported with the loss of only an embedded image, while all cell and data formatting were preserved. Importing a Lotus 1-2-3 file was not as successful. On a 123-97 file, all data was imported intact, but some formatting was lost—most notably the borders. On Lotus wk1, wk3 and Microsoft Excel 3 files, all formatting was lost, but the data was imported intact. All file formats preserved formulas in the source files, but all embedded images were lost.

Importing to Applix Presents *

All attempts to import graphics files failed. PowerPoint 4 files were a complete failure. The procedure actually started for PowerPoint 95, but failed. And the

biggest failure of all—Applix Presents does not even support the import of Freelance Graphics files.

Importing to Applix Data *

If you want to query an SQL database created with Oracle or Informix, or you plan to create a database from scratch, you might be able to use Applix Data. If you are thinking of importing an existing database, forget it because Applix Data will not even open an industry-standard Dbase III or IV file.

Exporting from Applixware ***

Applixware seems to export better than it imports. Using a text document with various paragraph and font formats, I was able to export an Applix Words document to HTML without any errors. It picked up underlining, bold and italics without a hitch and had no problem with justification.

I exported the same document to Microsoft Word 95/97 format (using RTF), again without a loss in formatting—even successfully exporting a forced page break. When exporting to Corel WordPerfect, the only error occurred on full justification, as it replaced *full* justification with *left* justification.

As for Applix Spreadsheets files, it exports basic font formats, data and simple formulas, but it drops embedded objects such as graphs when exporting to both Lotus 1-2-3 or Microsoft Excel formats.

Applix Presents files can be exported to Microsoft PowerPoint 97 format, but it seems to do only simple slide presentations well.

As a standalone productivity suite, the Applixware package is more than sufficient. It is inexpensive (I've seen it for as little as $159.00 Canadian), powerful, and in most cases, easy to learn. The major pitfall is the import issue in which it cannot properly import some of the more popular file formats, most notably dBase and presentation files. If you plan to use it to create files for the rest of the world, stick to the word processor and spreadsheet modules. You will be able to import and export to the most popular formats, if not in native format, then through RTF.

I deliberately did not review the Applix Mail module, because e-mail clients are a dime a dozen, and while it is a nice bonus, I would not purchase a productivity suite for it. I also skipped Applix Builder, since I am not a programmer and could not provide an informed opinion.

Personally, I feel some major improvements still remain to be made to Applixware. On its own, Applixware is a very good package, but to be adopted

in the business world, it must be better able to import and export other formats. In a perfect world, we would all be using Linux. But since we aren't in that perfect world yet, we still need to be able to play well with others.

Dean Staff is the manager of NovoClub, a computer bookstore in Ottawa, Ontario, Canada, and can be reached by e-mail at dstaff@echelon.ca .

Archive Index Issue Table of Contents

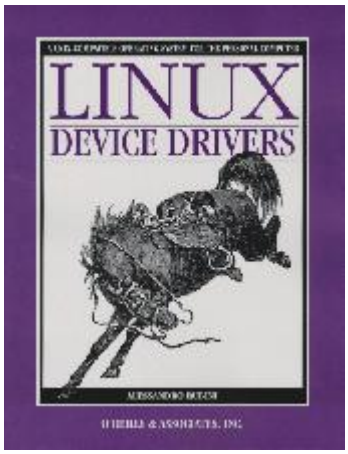
Advanced search

Linux Device Drivers

Mark Bishop

Issue #65, September 1999

Be forewarned, though; if you are not comfortable with C, you won't understand the examples that are amply spread throughout.



- Author: Alessandro Rubini
- Publisher: O'Reilly & Associates, Inc.
- E-mail: info@ora.com
- URL: http://www.ora.com/
- Price: $29.95 US
- ISBN: 1-56592-292-1
- Reviewer: Mark Bishop

In the last few years, we have seen an explosion in the number of devices supported by Linux. If you ever want to know how device drivers work or add kernel support for a particular device, *Linux Device Drivers* is the book for you. Be forewarned, though; if you are not comfortable with C, you won't understand the examples that are amply spread throughout. Also note that no matter how good a Linux device driver book really is, it will become outdated simply due to the rate at which the Linux kernel is developed. Mr. Rubini is aware of this fact, and throughout the book, he generously notes the

differences between kernel versions when applicable. He even goes so far as to devote an entire chapter to recent developments in the Linux kernel.

I found *Linux Device Drivers* to be very complete in its description of the Linux kernel versions for which it was written—2.0.*x* to 2.1.43, which were the most recent at the time of its initial printing.

This book is roughly divided into two parts. Part One includes chapters 1 through 10 and starts out with a simple “Hello World” module, then moves on to describe how kernel modules are properly set up. Part One continues through the entire gambit of modular device programming needed to write a full-featured driver for a character-oriented device.

Since the audience may not have any experience writing modular device drivers, a chapter on debugging techniques is included. I found these to be quite useful, and they helped me to accelerate the development of several kernel drivers I was writing. One thing I did find baffling is the order in which several chapters are presented. The one that stands out most is Chapter 10, which deals primarily with portability issues. This chapter could have been presented earlier in the book to help maintain the flow the author had sustained early on.

I found Chapter 9 to be the most useful and the most fun; it covers interrupt handling. To help facilitate the reader's understanding of how interrupts are processed and handled, it requires the reader to modify his parallel port by connecting two pins together. (The parts can be found at your local computer hardware store for about $3.) Once my parallel port was modified, I was able to make full use of the examples. Mr. Rubini takes a difficult subject and breaks it into manageable parts. He does this quite effectively, moving through very technical topics with great fluidity.

Part Two of *Linux Device Drivers* covers more advanced topics. These include block drivers and network interfaces, and how one would write device drivers specific to them. Part Two also covers memory management and device access on peripheral buses such as the PCI and ISA bus. It also delves deeper into topics briefly mentioned in Part One.

Also in Part Two is something I have rarely seen described in other Linux kernel books—a description of the actual physical layout of the kernel source. Mr. Rubini approaches this by following the path in which the kernel boots from the first architecture-independent function (**start_kernel**) through the **init** process. This is also one of those chapters which might have been better placed earlier in the book, but no matter where it's located, its inclusion is greatly appreciated.

Linux Device Drivers serves many different purposes. This book will always serve the developer who wants to expand the number of devices which work with Linux. Also, you need not have the desire to write a device driver to learn something from it. All you need is an interest in knowing more about how devices work with the kernel to provide the services and stability we want. It is a great piece of work, where the one major drawback is time—this book was published in February of 1998. Changes made in subsequent printings may be found at http://www.oreilly.com/catalog/linuxdrive/errata/. I urge Mr. Rubini to follow up with a second edition, and I plan to be first in line for my copy.

Mark Bishop (mark@bish.net) recently graduated from Southern Illinois University and moved to Tampa, Florida, where he now has a job in the engineering field. He is primarily interested in developing embedded applications. Whenever he can, he works on the MP3 player he designed for his Jeep.

Archive Index Issue Table of Contents

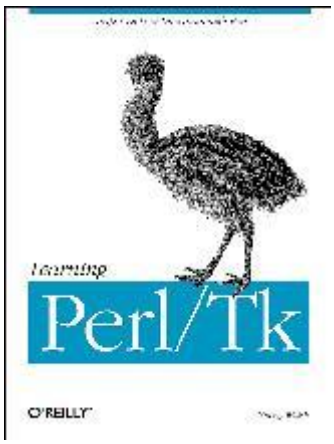
Advanced search

Learning Perl/Tk

Bill Cunningham

Issue #65, September 1999

This book is targeted for those just beginning windows programming in Perl/Tk on both UNIX and Windows platforms, although veteran programmers will find a wealth of detailed information in it as well.



- Author: Nancy Walsh
- Publisher: O'Reilly & Associates
- E-mail: info@ora.com
- URL: http://www.ora.com/
- Price: $32.95 US
- ISBN: 1-56592-314-6
- Reviewed by: Bill W. Cunningham

Learning Perl/Tk, by Nancy Walsh, is the first of what will undoubtedly be many books on this subject, and it will be a tough act to follow. This book is targeted for those just beginning windows programming in Perl/Tk on both UNIX and Windows platforms, although veteran programmers will find a wealth of detailed information in it as well.

Learning Perl/Tk is well-written and thoroughly researched. It begins with a brief history of the Perl and Tk languages. It tells exactly where to download the Perl/Tk module, how to unpack and install it, and how to test your installation to see if you are ready to write Perl/Tk programs.

Next is a general discourse on window geometry, and the concepts of the widget and the event loop. These are details the windows programmer must deal with that do not apply to the traditional Perl programmer. Ms. Walsh does a fine job of explaining these concepts in terms anyone can understand.

The majority of the book is devoted to a detailed description of the most commonly used window widgets: the button, check box, radio button, label, entry, scrollbar, list box, text box, canvas, scale, menu, frame and composite widgets such as the dialog box. Virtually every aspect pertaining to these widgets is covered in the minutest detail: color, placement, size, border size, style and functionality. Other widgets are available in Perl/Tk, but these are the most common ones, and in order not to get too far out in left field, the book limits its focus to these.

The author's code examples are written in an elegant, concise, easy-to-follow style. For example, the following code creates five text entry widgets, each in a different relief style, with the style's name displayed in the respective boxes:

```
foreach (qw/flat groove raised ridge sunken/) {
$e = $mw->Entry(-relief=> $_)->pack(-expand=> 1);
$e->insert('end', $_);
      }
```

This example appears on page 112 of the book. Throughout, the author demonstrates an enviable command of the Perl language, particularly the object-oriented features.

The book does a great job of bringing the first-time Perl/Tk user up to speed. By the end of the second chapter, the reader will have at least five fully-functional windows programs up and running. And for the veteran programmer, the book goes into a level of detail about the various window widgets which should meet the most demanding needs.

There is something of a void of information for the intermediate-level programmer. For example, there is no discussion of how to create a text-entry widget, get some user input from the widget, and use Perl to do something useful with the input. The Perl/Tk module does come with many demo examples that basically fill this gap. I did notice a few minor typos in the book, but it was obvious even to me (a non-programmer) how to fix them. Almost all the code runs perfectly. I know, because I typed in and ran each example!

Virtuous (i.e., lazy) programmers can download all the book's code from the O'Reilly web site (http://www.oreilly.com/catalog/lperltk/).

Learning Perl/Tk provides a simple yet thorough introduction to this newest method of writing windows programs, and will also be a valuable reference for the production programmer. There is room for more books on this subject at the intermediate level.

I believe anyone with an interest in Perl/Tk will benefit from reading this book, and I recommend it with enthusiasm.

Bill W. Cunningham, Gunnery Sergeant in the U.S. Marine Corps, is a system administrator with the Second Marine Aircraft Wing, G-7, Cherry Point, NC (the greatest job in the universe). He likes playing guitar, reading, driving and spending as much time as possible with his wife and four kids. He can be reached at bwc@coastalnet.com.

Archive Index Issue Table of Contents

Advanced search

Letters to the Editor

Various

Issue #65, September 1999

Readers sound off.

tkdiff

This is in reference to “A Toolbox for the X User” by Christoph Dalitz, May 1999. In it, Mr. Dalitz mentions the “tkdiff” utility, but seems to base his observations on an old version. I wanted to alert you to a couple of significant changes, which may impact tkdiff's usefulness to some users.

Thanks to much work by Bryan Oakley, 3.03 (the latest released version) contains many features and enhancements relative to the more ubiquitous 2.*x* and 1.*x* releases.

One change in particular addresses the author's observation that “tkdiff can be invoked only from the shell prompt because it requires file names as command-line arguments.” This is no longer the case. If tkdiff is invoked without arguments, it will pop up a dialog box and prompt you for file names (via text-entry widgets coupled to optional file-selector widgets). As such, tkdiff is now a suitable candidate for being launched from a toolbar or similar mechanism which eschews the command line.

Relative to the observation that tkdiff has “no external man page, which occasionally makes it inconvenient to get usage information”, I would like to point out that the on-line help does come up automatically, if invalid command-line parameters are used. This is still inconvenient, but better than having to explicitly invoke the on-line help system in order to see the help text.

Lastly, on a related note, I've turned over maintenance responsibilities for tkdiff to Damon Poole of Ede Development Enterprises. Damon graciously volunteered to maintain and enhance the code on an Open Source basis. The

official tkdiff home page is now at http://www.ede.com/free/tkdiff/index.html, from which this 3.02 release can be obtained. Thank you.

—John M. Klassa klassa@aur.alcatel.com

Bug in Pthread Code

There is a nasty bug in Listing 1 published in “Introduction to Multi-Threaded Programming” by Brian Masney, May 1999. The line

```
pthread_t *tide:
```

should be changed to

```
pthread_t tide[10];
```

—Klaus Pedersenklaus.pedersen@nokia.com

Coming of Age

I have followed Linux for four years and wanted to voice my support and thoughts. I was prompted to write after reading the June *Linux Journal*. It strikes me that Linux is at a crossroads of sorts, where identity and maturity need definition and direction. This point was made in the articles on Linux standards and were reflective of the development community and its needs. However, there is another perspective, that of the consumer or end user.

I consider myself to be in this latter community. Labeled as such, I do not believe that the Linux community has come to terms with an identity or maturity. As an end user, I am amazed at the wealth of information and capability that has been developed to date. In many ways, this reflects very well on the Linux community and should be a great source of pride—well done. Nevertheless, the question is where to go from here. The “here” is subject to debate; however, I would argue that the “here” is further adoption by end users like myself.

We are all well aware of Linux's success as a development platform, Internet web server, enterprise mail and file server, and specialized supercomputer platform. Unfortunately, these applications represent a very limited set of users. What about the end user who operates in the world of Microsoft today? These users have very different needs than those of a development community. These users typically see the computer as an appliance. The computer does things that provide benefit to the user. This type of user sees the computer like most people see an automobile—get me from point A to point B, but don't ask me to understand how it happens. I can sense some developers community taking great pride in using a tool that requires great skill

to master. Unfortunately, end users, for the most part, are not interested in acquiring those skills.

I'll use my own experience to illustrate the point. I've gone from a distribution of Linux by InfoMagic in 1995 through Red Hat 5.0, 5.2, Caldera OpenLinux 2.2, and a soon-to-arrive distribution of Mandrake 6.0. In all of this, I have not yet abandoned my Windows. This reflects two important limitations of Linux for me. First, the lack of user-friendly applications, and second, a lack of confidence in using the operating system. Limitation one is becoming less important with the maturity of KDE and the availability of Netscape, WordPerfect, StarOffice and Applixware. This fall's release of Corel Office should be a watershed event. Nonetheless, limitation two is still daunting.

As I write this, I have just returned from the bookstore with *Linux in a Nutshell* and *Learning the bash Shell* from O'Reilly. Thank God for O'Reilly! These books and others are part of my growing Linux collection as I try to understand what the system does. For example, why does the installation of Metro X cause conflict with the X server as distributed by Caldera? Why do I have to apply a print offset correction to produce centered text from WordPerfect? Why does the configuration of printers using Caldera's COAS seem so difficult and problematic? Why, when I follow the instructions for building a new kernel, does the kernel not work? These and other problems have haunted me as I try to use Linux from an end user perspective.

My point here should be obvious and simple: these types of issues are not a concern when I use Windows. If the Linux community wants to see greater user adoption, then the community needs to think like an end user. Trust me in saying that I am an activist end user, having spent hundreds of hours reading and learning. However, unlike me, most end users will expect simplicity, reliability and utility. Linux today can deliver reliability; simplicity and utility are not part of the equation. To put it in perspective, if my mother asked me to recommend a computer, my answer would be a Wintel machine because I could walk her through most problems over the phone. With Linux, that would not be possible.

In sum, I think the Linux community needs to assess which direction it wants to go in—remaining a specialist operating system supporting networking and unique applications, or becoming more of an appliance operating system that users can operate confidently. The time to solidify an identity is now, and the opportunity to come of age will always be timeless.

—Mark H. Runnels mrunnels@home.com

Great awk Article

What a pleasure to read Mr. Iacona's introductory treatment of **awk** in the June issue. I've been exposed to awk in a hard-core UNIX setting (designing SRAM circuit layout for Motorola), but have not used it myself. Now I design textiles on the SGI platform (not so far from chip layout as you might think) and need to manipulate text files. I run LinuxPPC on my PCC PowerCenter 120 Macintosh clone at home to practice UNIX for work. I'll lean on Mr. Iacona's article, his clear, well-commented example code and helpful glossary with confidence. I'm dying to get the O'Reilly books and take on more. A clear, usable, focused treatment of a daunting subject; well-edited, too. Continue to set your standards high.

—Four Hewes four@bway.net

Kernel Korner

The information in this month's Kernel Korner (“IP Bandwidth Management” by Jamal Hadi Salim, June 1999) reminds all of us that there is yet another dimension to the “free-ness” offered to Linux users: not only the code and applications in development, but also Linux ISP's.

These ISPs are affecting our ability to move *free*ly. Why? QoS (Quality of Service) is essentially the end of all-you-can-eat Internet access—no more monthly flat rates. Oh sure, there's the bare-bones rate, but that's *bare*, *bare* bones. Instead, the more we are willing to pay, the more our ISP will shove down our lines. What a fair world!

It's a double-whammy. I can remember the good old days (i.e., three years ago) when we criticized telco's for ripping us off (after all, the customer is considered “the last mile”); now, ISP's will also be subject to our scorn (if they're not already), as will the host client/servers (i.e., those Linux boxes we adore so much).

Sure, QoS implementation is not so much a question of “if” but “when” (it's *that* inevitable, my friend). But my teeth gnash at the thought of this implementation hard-coded into the Linux kernel.

—John K. Joachim Joachimj@usa.net

Rave

I've been a subscriber for a few years but still consider myself a newbie. So far, my faltering steps toward ever greater Linux mastery are consistently rewarded by finding out just how rich and robust an OS Linux is. Your magazine really

helps; it's great to feel like I'm part of such a community as *LJ* serves. Thank you and your staff for sweating the little stuff.

—Jamie Matthews mattja@iglobal.net

Archive Index Issue Table of Contents

Advanced search

More Letters to the Editor,

Smudgy Ink and Page Borders

I got the August issue yesterday, and the letter regarding the black borders on the Feature pages caught my attention. Having worked here and there in the publishing business, I feel very qualified to offer the following alternatives: Instead of using a solid black border, why not use a 25% border with the screen angled at 60%? Less ink, and it still makes the pages stand out when looking at the edge of the magazine. Another possibility is to go ahead and keep a solid black border, but only along the top of the pages. That way, when the reader is reading it, there isn't the ink on the left and right edges where the magazine is held. Still visible from the “outside”, just not all the way around.

Keep up the excellent work. (I know you will anyway, since Linux users tend to be the type not to put up with a bunch of PR bull.)

—Mark, gus3@bright.net

BTS

In the August 1999 issue, #64, one Lisa Zuckerman (blueink@netzero.net) wrote in asking why KDE was failing to find “libstdc++2.9”, as well as asking why vi was telling her that .xinitrc was not a regular file. The answer given by Mr. Marc Merlin (marc@merlins.org) was rather incorrect, and not very helpful. So, if I may, here is the /correct/ answer...

1. Libstdc++ This is the standard C++ library commonly used in Linux, often it is supplied by GCC. Version 2.9, however, is supplied by egcs 1.1.X, which is not included with Slackware3.6, but IS included with Slackware4.0
2. ftp://ftp.CDROM.com/.4/linux/slackware-4.0/source/xap/kde contains all the necessary files, and a “kdebuild.sh” script, needed to build and install KDE on a Slackware system.
3. Switching distributions is not really necessary.
4. About .xinitrc

First, check the permissions on the file

```
ls -col ~/.xinitrc
```

Here is mine:

```
-rwx------   1 jeremy        552 Nov  5 15:40 .Xclients
```

you should see something similar, most important is the “rwx” at the beginning, that means that the owner can “r”ead, “w”rite, and e“x”ecute the file.
Second, if the permissions are correct, try this

```
file ~/.xinitrc
```

Which should produce results similar to the following:

```
/home/jeremy/.xinitrc: ASCII text
```

If it doesn't, then it has surely become corrupted. If this is the case, back it up, then create a new one (This example assumes you wish to start KDE by default)

```
mv ~/.xinitrc ~/.old.xinitrc
echo "exec startkde" > ~/.xinitrc
```

—Jeremy Crabtree, crc3419@delphi.com

Thinkpad 750CS Article

I was glad see the article in the March issue with information on the 750CS. I bought a Thinkpad 750CS for my wife last week, but had a great amount of trouble with the floppy. I ended up buying another laptop (a 755CSE) that had a 1.44MB floppy in it and put the hard drives in the 755 during my Linux install to get at a 1.44MB floppy. Thanks for the info on the floppy. I thought I had a useless 2.88MB disk drive until now.

I've since put a Raytheon Wireless LAN card in both systems to make mobile web terminals. Very nice, I can go to the pool and surf the web or reach the other laptop if it is nearby.

I have a few question though. One is _when_ to run tpdualscan... I can get it to work if I run tpdualscan manually then startx or xdm, but when I try to put it in my /etc/X11/xinit/xserverrc or rc.boot it runs but X looks the same as before.

Also, any luck with the on board sound? I identified the chip set as an mwave, but have had no luck in getting it to work.

BTW, I got the 750CS for $99 and 755CSE for $298 at a local refurbishing shop. I love these old laptops, and they work great under Linux... not too many platform can run on a 486 and perform so well. ;) Again, thanks for the great article, and keep up the good work.

Chris McClimans, chris@mcclimans.net

Opinion: LJ and the future

Hi, I'm a new *LJ* reader. I like your magazine but I have the following comments (and I hope, some helpful tips):

- The feature articles are much too techy; this month's issue I read everything EXCEPT the features. Linux is a great product and has some fascinating uses but, unless you want your target audience confined to R&D types in Universities, most people will be turned off. SUGGESTION: Have more features related to real-world solutions that a greater audience can relate to. ie. Business and consumer oriented features.
- Not enough product hype: Your mission statement is to promote the use of Linux so why not add some excitement to it? Pit products against one-another, show performance benchmarks, etc. Advertisers love this stuff. You don't have to be as unabashedly propagandistic as some Windows oriented mags but adopting some traits from a proven formula would help. Hype gets consumers/ readers excited about products and that in turn gets advertisers/ vendors excited about your magazine. SUGGESTIONS: More articles showcasing specific products: tone of articles in general should be more “pro-consumption” (ie. You NEED to upgrade) & not too techy;

Now I realize this probably isn't news to you and that you are caught between keeping existing readers happy with the current format and trying to gain a bigger market with a different pitch. However, the Linux phenomenon is growing all the time and I believe more and more consumers will want a magazine with the aforementioned characteristics. The challenge for The *Linux Journal* is to inevitably evolve along with the Open Source Revolution. I feel that it is perfectly positioned to be THE AUTHORITY on Linux product reviews ala PC Magazine in the Windows world.
In closing, I hope that my comments are helpful even if you don't necessarily agree with them. Here's hoping *LJ* becomes a big success!

—S.McPherson, smcpherson@microworks.a

“Letters to the editor”

After visiting the Red Hat, SuSE and Debian web sites, I realized that everyone seems to avoid posting any graphic icons of “tux” the Linux penguin.

Its interesting how commercialization of Linux can cause people to avoid using the primary Linux logo (tux).

An interesting article for *LJ* might be “Linux Marketing Strategies & Tactics” or something similar...

I think if we stop and take a look at how Linux is being marketed, we would see disturbing trends that are emerging everyday..

Larry Ewing, the creator of the tux logo has a web site:

http://www.isc.tamu.edu/~lewing/linux/index.html

—Shiloh Costa, ap296@torfree.net

LTE letter

George Saich
—*Cheers*+ for your well-considered essay in the *LJ* e-mail column! While not out of place, your apology for your “extreme” stance earns my verbal support.

It is my belief that Extremism in the Persuit of Computing Freedom is no vice. I would suggest that something like California's Community College System should be recruited to provide this training (trainers need State Instructor's Certification, and students gain a modest State subsidy) at minimal end-user expense (and the “required manuals” for courses should be LDP-standard downloadable documentation) in keeping with the Free Software Spirit.

Do other states have Community College, continuing-education-type, public-access education systems set up? Here in the SF Bay Area, anyone can take a good, 5-unit college-credit training course for one quarter for well under a hundred dollars. Similar training courses (taught by private training groups) cost _many_ hundreds and often cram the whole session into three weeks, usually requiring students to take time off work. The difference between these two approaches is bloody obvious!

—Eric Palmquist, epalm@blueneptune.com

Article Review for issue 2!

Article: Linux System Administration: You as System Administrator, Issue 2

Mark:
I just read your 3-pager on System Admin' on the net. I sincerely enjoyed your style of writing.

As an educator in this industry for over 15 years (CNI/MCT, others) I pride myself in the ability to “never have loose scope” of the first-time users perception of a particular subject matter,... especially one of this nature and magnitude. The analogies used to “link” the concepts of the Unix/Linux file system using the DOS environment were excellent. Thank you for publishing this on the net!

I am currently evaluating other such documentation as well as co-developing as series of courses in Linux (re-inventing the wheel to “my” way) and look forward to reading more of your work...Again..Thanx!

—Tom Foster, tfoster@frostbank.com

Article: "The Point Really is Free Beer"

I am astounded that your usually top-class publication has included an editorial such as that named above in your July issue of *Linux Journal*.

To my reading it is extremely inflammatory, making sweeping general statements that lack an appropriate level of reasoned or logical exposition (which I would normally expect of your magazine). It appears to be (in my opinion) one person's petty ranting about his feelings on the GNU General Public License (GPL), shrouded in flamboyant “high moral” talk of public good.

I guess the title really sums it up - to Mr Hughes, the most important thing is “no cost software” as opposed to software guaranteeing the (long term) freedoms and rights of an individual to be a moral citizen (as exemplified by the GPL license).

Of course, it is up to Mr Hughes to show his true colors (which I believe may well be bright) when his company's “soon to be announced” first products and services are released. If he “has been worried about software infrastructure ever since” I suspect he may well offer something wholesome to society. I am however sceptical, based on the apparent contradictions (or perhaps just misunderstandings) in his editorial.

The proof will be in the 'beer' that he offers. If it is just 'beer', ie. software provided at “no cost”, yet restricting my freedoms as a moral, caring member of the human race, I personally will not touch it (nor will any of the 25-odd employees under my management at my 'development company' (to use the same petty legitimizer)). However if Mr Hughes' ultimate intentions are positive and good, I will be the first to support his work. Certainly I have heard only positive information about the Cypherpunks, of which he was a founder. We could perhaps all take heed not to trip over our own exalted egos (yes, myself included).

Please put your “editorial advisory board” (as listed on page 4) to work in ensuring the top quality content usually expected of your magazine.

Sincerely (and a little (in)“flamed”, and surprised),
—Zenaan Harkness, zen@getsystems.com

Credibility

I am rather disturbed by several articles that appear in the May 1999 issue. If *Linux Journal* continues to publish programming articles by technicians who work “at a local computer store” or book reviews by high school students, I won't be renewing my subscription again. The first case is “Introduction to Multi-Threaded Programming” by Brian Masney. A quick look at the sample code provided is all it takes to see what kind of “authority” Mr. Masney is. Besides the uninitialized pointer in main() that produces an immediate segmentation fault it is quite clear that this guy never learned a thing about neatness. Publishing this kind of code will never do anything to promote readable code. As for “Review: Linux Programmer's Reference” by Richard Petersen I am completely

dumbfounded as to why reviews are being handled by high school students. I am not trying to denigrate Mr. Petersen's knowledge or ability but I expect articles in this magazine to be written by authors who are more experienced and can provide a viewpoint that is more relevant to those of us who have spent years in the trenches on major league projects. If *LJ* can't come up with contributions by more credible authors I don't think that I will be the only one deserting. Just my 2 cents.

—Bill Lewis, satan@ns.net

Beginners section

With all the new people looking into using Linux, I recommend that *Linux Journal* add a larger section for the novice Linux user. If you keep this section separate from he rest of the magazine, the experienced Linux users can easily ignore it. However, it would be very beneficial for new Linux users.

—joe lerch, ljr@globalfrontiers.com

Issues with the August issue...

First, my compliments on a nice magazine. I've been a subscriber since the second issue. I have always appreciated the clean approach you have taken with the publication.

That said, I'm a bit disappointed at the direction *LJ* seems to be taking. The first thing I noticed was the foldout advertisement for Penguin Computing. ARGH! I thought *LJ* would remain pure. I detest the standard computer rags for the use of reply cards and “fat ads”. The first thing I do if I ever read one of those magazines is rip out all the cards and ads. I suppose this is an economically driven decision to keep subscription prices down, but please don't do it. I would rather pay an extra couple of dollars per year to avoid this annoying, wasteful, scourge :)

It seems the layout is moving to that of those “other” publications as well. Your “Up Front” section is difficult to follow. Trying to forge a path through a myriad of sidebars, arrows, boxes, etc., makes it difficult to pick out the important information. This practice doesn't seem to have infested the main part of the journal (the articles) and for that I'm grateful. Please limit the font-o-philia and layout fluff to the the front few pages. It felt like I was watching MTV with frequent cuts, fast pans, and no content. Ouch! (A minor beef: the lengths of the bars on the bar graphs depicted do not correlate with the numbers indicated. Confusing.)

Also, I second a reader that noted the smudging of feature articles with the thick black border. Perhaps some of us excrete solvents through our thumbs that are effective in dissolving the ink used in *LJ*?

My last comment is on the “Best of Technical Support”. I'm rather dismayed at the level of the responses. Many of the questions chosen for printing are fundamental newbie questions, but often the responses are

cryptic, cursory, and probably do more harm than good. Not that I purport to be an expert, but these responses should be screened for accuracy and completeness before being printed.

Ok, enough griping. I applaud your work and will continue to read *LJ*. Just please remain true to the clean, informative approach. I'm sure you never hear from the silent majority that has no pet peeves worth complaining about!

—Steve Singleton, ssinglet@coe.edu

A Letter for the "Letters" Column

I must respectfully disagree with Troy Davidson in his letter, “Standards”, of the August 99 issue.

As to the HP 722c printer, I dare say it can be made to work with any Linux distribution. I use an HP 672c at home with Red Hat 5.1, and these 2 printers are fairly similar. Try www.experts-exchange.com, and I wager that within 2 days you will have info on exactly how to fix that printer. This web forum is tech support beyond anything commercially available.

I don't personally see the Linux community as trying to beat or destroy Microsoft (at least not overtly). Linux appears to me to be a collective effort to produce the best operating system and associated applications possible. The motivation behind this effort is more of an artistic challenge than an economic target of opportunity.

If Linux is better than NT and Win98/5, and I believe it is, then it's only a matter of time until it wins over the market share on its own merits—without a declared war against Microsoft.

Respectfully,

—Bill W. Cunningham, CunninghamBW@2MAWCP.usmc.mil

Linux Expo

Hello, my name is Dave and on the 24th of may I sent the message below to the Linux Expo coordinators. I still have not received a reply and I am still disgusted at the unfair treatment I received at the expo. Since the expo, I have stopped using Linux, stopped promoting Linux, and switched to OpenBSD. I feel Linux as a whole is turning into a huge moneymaking scheme just because it happens to be the popular “flavor of the day”. I will continue my subscription to *Linux Journal*, in my opinion, it is simply the best computer magazine in publication. Thank you for your time.

> Sent to linuxexpo.com on 24 May 99—Hello, I attended the Linux Expo this weekend with my two daughters ages eight and seven. You can imagine my shock when I was asked to pay full

> admission price for them. How can you expect to charge for people who don't understand anything that's going on inside? I paid the price, because I wanted to attend but was unable to purchase anything as laying out the additional $40 for my two daughters cleaned me out. I was very upset by this and DO NOT plan to attend next years event. Unless, of course, I am given a refund of the $40 and you change your policy of charging full admission for children.

—David J. Pote SSgt USAF, david.pote@seymourjohnson.af.mil

Article in August Linux Journal

I was disappointed to see that your article “Graphical Toolkits for Linux Programs” overlooked the very powerful and free Fast Light Tool Kit (FLTK). FLTK is an excellent GUI widget set with corresponding GUI builder that works natively with both the win32 and the X/11 API. It also extensively supports OpenGL. The widgets are very high quality, fast and compact. The whole library is released under the LGPL, which makes it a much better choice for commercial development than Qt or Motif.

Check out http://fltk.easysw.com for more information on this excellent, free GUI widget set and GUI builder. Sincerely,

—Darren Humphrey, dhumphre@simulation.com

Multi-Threaded Programming

I have a comment regarding the good introductory article on page 74 of your May 1999 issue.

Mr. Masney states “...threads...will automatically take advantage of machines with multiple processors.” Based on my experience, this is not [yet] true.

I used pthreads to design & code an image resampling benchmark. I ran this program on a 14-processor SGI (IRIX64 V6.2) & on a 12-processor Alpha (OSF1 V4.0). In both cases vendor-OS specific function calls had to be added to ensure each thread executed on a different processor. The SunSoft document Multithreaded Programming Guide indicates the same is true for Solaris 2.4.

In conversation with systems analysts at the facilities I used, I learned that the POSIX pthread design did not include standards for implementing multi-processing. I do not know if it has since been updated.

I initially developed the benchmark algorithm & program on my 486/ Linux desktop (no threads). I ported the code to both SunOS (no threads) & Solaris (pthreads) inside of an hour. The code quickly ported to the SGI & Alpha requiring minimal consultation with system analysts. The best

performance was achieved using vendor thread libraries. Conversion from pthreads to vendor threads required no major code alteration.

—Robert Geer, bgeer@xmission.com

PCI Modem letters-to-editor

In response to Greg Bailey's correction concerning SOME PCI modems being Linux-friendly, I feel compelled to point out that the the problem lies in “WinModems” - modems designed to operate only with MS Windows, through a LICENSED INTERFACE to MS Windows. These modems are often bundled with systems or sold very inexpensively. The fact that 3Com has a “generic” 56K PCI modem with a street price of $100 and a PCI “LoseModem” for only $60 makes me wonder what incentives may exist in the marketplace for making Windows-only modems.

Thanks for the great magazine!

—Peter Cavender, cavender@sover.net

PS/2 mouse problems FYI

I have installed them both, and have found a common problem in both. I think it is a kernel problem....

I have a Compaq 5630 Presario with 1 serial com1 port, 1 ps/2 mouse port, a parallel printer port, with plenty of memory and a quantum bigfoot noisy as hell ultra IDE drive.

I also have an OEM Compaq branded 2 button ps/2 mouse , and a 3 button MouseSystem dual-mode serial/ps/2 “white” mouse both mice work perfect in “all” modes of operation both serial and ps/2 on my windows partition. Same can not be said of my Linux partitions.

I began by installing Red Hat with my MouseSystem mouse on the serial port. This was a good and lucky choice, as I was later to discover. After installation it occurred to me that I might need to use my serial port for something else, so I tried switching my MouseSystem mouse to the ps/2 port and changing my config to use it. Dead meat city, as soon as I tried to use the mouse on the ps/2 port, the entire keyboard locked up and my system was dead. This was Red Hat 6.0.

Later when I tried to install caldera with the fancy lizard install which tries to detect the mouse, I had the same problem.

This occurs also with my Compaq branded 2 button ps/2 mouse.

There are a ton of problem reports in the caldera user-forum archives about problems with ps/2 mice.

One person made the observation that the “paraport” driver was reporting the detection of a “ps/2” device... This would seem like a big red flag to me shouting KERNEL PROBLEM !!!

I don't understand why caldera does not pass these problems along to somebody who works with the kernel or its drivers, so that it can get fixed.

Could you please notify somebody in Kernel land, that we are having a problem out here in the wilderness

—Steve Bovy, Steve.Bovy@sterling.com

“PCI modems”

I sure thought by now someone would have pointed out the problem, but from the “Compounding Errors” letter and response, it doesn't look like it.

All you said about PCI modems would have been right, if only you had used the term “Winmodem” instead. I don't doubt that some (or maybe even all, I never saw one myself) Winmodem actually uses the PCI bus, but that's rather irrelevant to the problem.

A Winmodem is a modem that doesn't include most of the CPU and software needed to do anything decent; instead, it has software (typically only for Windows) to get the host CPU to do it instead. This supposedly makes for a cheaper modem. Unfortunately, you pay for it by having a chunk taken out of your main CPU, but hey, we all know Windows users don't *really* multitask anyway, right? And nobody needs docs on how to do this for other OSes, because Windows is all there is, right?

One guy recently claimed on linux-kernel to have access to the necessary docs and source to write a driver for one specific Winmodem for Linux. I have no idea if anything will ever come out of this, however.

Oh, and there also seem to be Winprinters around with the exact same problem, I hear.

So: please don't blame the PCI bus for this!

—Kai Henningsen, kaih@khms.westfalen.de

Archive Index Issue Table of Contents

Advanced search

UpFRONT

Doc Searls

Issue #65, September 1999

LJ Index, Strictly On-Line and more.

LJ INDEX—September 1999

- Speed achieved by the 350-node Cplant98 cluster running Linux at Sandia National Laboratory: **125.2GFLOPS**
- Where 125.2GFLOPS places the Sandia system in the current TOP 500 list of supercomputers: **53**
- Position of microsoft.com among the top sources of visitors to the new linux.com: **1**
- Number of microsoft.com visitors in the first two weeks of linux.com's operation: **15,000**
- Total age of Phat Linux's two founders: **30**
- Number of Net-connected computers whose spare CPU cycles are devoted to searching for extraterrestrial intelligence by SETI: **625,253**
- Total CPU time of all those computers: **99,799,890 hr 45 min 38.8 sec (11,392.68 years)**
- Number of "results" returned by all that terrestrial intelligence: **2,258,824**
- Percentage of those results produced by Linux platforms: **12**
- Position of Linux among all platforms in results performance: **2**
- Professional attendees at Linux Expo Paris 99: **5000**
- Number of exhibitors and vendors at Linux Expo Paris 99: **87**

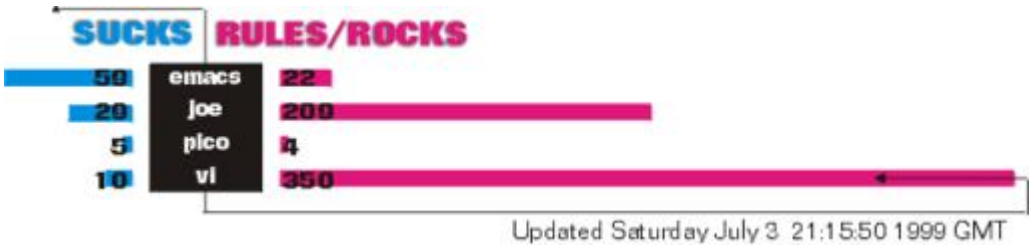
Sources

1. Phat Linux, www.phatlinux.com/about.html
2. VA Linux Systems, http://www.valinuxsystems.com/
3. TOP 500, http://www.top500.org/

4. *Linux Today*, www.linuxtoday.com/stories/6797.html?nn
5. SETI site, June 15, 1999, http://setiathome.ssl.berkeley.edu/
6. Linux Expo Paris press release, June 23, 1999

ROCK & RULE

vi rocks. It also rules. So says Vassilii's Editors Sucks-Rules-O-Meter, which mines the verbs on Altavista and pronounces vi the winner over Emacs and all the other editors as a subject of those two superlatives. As of July 3, vi had a 350/10 rules/sucks ratio. And it looks like Linus isn't the only one out there who hates Emacs—a sentiment he shared with us on a recent panel that also featured Emacs creator Richard Stallman. Emacs' rules/sucks ratio is 22/59. This meter can be found at http://www.tarunz.org/~vassilii/srom/and is updated weekly. Thanks to Vassilii Khachaturov.



BARREL SCRAPINGS

Want a new domain name? Good luck. We are at the bottom of the .com barrel, and the .net and .org barrels must be getting fairly low too. While the urban legend says every word in the dictionary has been sold for .com use, “misstep” is still there. So is “dodder”. A lot of two-word combos (such as hunkerdown) are gone as well, but a few (such as stupiddog) are still there. But your chances of getting the domain you want are being reduced every second by the sharks who buy domains from Network Solutions for $70 and then sell them for far higher prices (up to millions of US$) to the unfortunates who came too late to buy direct.

Thus, your only two strategic naming choices are anonymous or strange—or both. Why not create a front company with a camouflage name like “Symnetix.com”, while your actual business will be an enterprise NT replacement service called “Bizfloss.com”? If you do that, remember who your friends are when you file for that IPO (initial public offering).

To save you a bit of work, I went through the familiar **whois** routine to scope out the possibilities. They are mighty slim. Let's say you are in the bug zapper business and want “bzzt.com”. Well, Allan Henning of Stockton, California has already grabbed that one. How about dropping a z? Nope; “bzt.com” belongs to Hovinga Holding in the Netherlands. How about adding a z? Wrong again; “bzzzt.com” has gone to the Mikluhomaklai Sensation Corp. in Omsk, Siberia.

Okay, how about one more z? Voilà! You can have it. Now prepare to spend the rest of your business life saying, “that's bzzzzt.com with four Z's.”

- awfulstrange.com
- glassgargle.com
- spizanch.com
- girlfriendfromhell.com
- loadolinux.com
- thirdwife.com
- rsh2god.com
- yerassismine.com
- pantywad.com
- tuxdisk.com
- niddle.com
- stoptalking.org
- stopmakingsense.com
- thunderingnerd.com
- 2manyservers.com
- nowgoaway.com
- stilldead.org
- avatato.com
- pocato.com
- thbzzz.com
- birdgrinder.com
- crosslips.com
- isobaptist.com
- umess.edu
- bozoretentive.com
- fubar.mil
- overdive.com
- deathpatty.com
- halfcat.com
- twitchingannoyances.com
- zonecontact.org
- flatoutbad.com
- placentamix.com
- hatefuljavapages.com
- java2die4.com

- aaaarg.com
- obytheway.com
- freedeath.com
- buildingcozy.com
- condomcrobar.com

—Doc Searls

LINUX USERS

According to International Data Corporation (IDC), there are more than 10 million Linux users worldwide. In 1998, the Linux market grew by 212 percent, and nearly 18 percent of all server hardware licenses sold last year were Linux, according to IDC. Another consulting firm, NetCraft, estimates that Linux or other Open Source software currently runs on more than half of all web servers worldwide.

Some interesting companies/organizations using Linux include:

- Boeing
- Mercedes-Benz AG
- Yellow Cab Service Corporation
- Canadian National Railways
- United States Postal Service
- National Disaster Communication Response Team
- World Council of Churches
- Sony Electronics

Source: Mercury Information Technology, Inc., http://www.m-tech.ab.ca/linux-biz/

STRICTLY ON-LINE

Adventure by Joseph Pranevich is a nostalgic look at the old “Colossal Cave” game and its various iterations—a fun game and a fun article. Mr. Pranevich tells us a bit of his and the game's history and how to play it.

Remotely Monitoring a Satellite Instrument by Guy Beaver is the story of a small aerospace company involved in a NASA-funded satellite mission to study the atmosphere. A major portion of this experiment involves calibrating and testing the instrument. While done on the ground, the calibrations are monitored remotely using a Linux-based system. Most of the software used

was originally Windows-based, but has now been ported to Linux to take advantage of the many open-source products available.

First UNIX/Linux National Competition Held in Ljubljana, Slovenia by Primoz Peterlin and Ales Kosir introduces this competition and the winners. They also present both the problems and the answers that made up the test.

Linux Apprentice: Filters by Paul Dunne gives instructions for simple data manipulation commands in Linux. Covered commands include grep, egrep, tr, sort, head and tail. Mr. Dunne also takes a look at programmable filters such as sed and awk along with their use with pipes.

A book review of **The Unified Modeling Language User Guide** by Geoff Glasson. If you are a programmer involved in producing object-oriented software systems, you will want to know how this book can help you.

EVENTS

- LinuxWorld Conference & Expo, http://www.linuxworldexpo.com/, August 9-12 in San Jose, CA.
- O'Reilly Open Source Convention, http://conferences.oreilly.com/, August 21-24 in Monterey, CA.
- 8th USENIX Security Symposium, www.usenix.org/events/sec99, August 23-26 in Washington, D.C.
- 3rd annual Atlanta Linux Showcase, http://www.linuxshowcase.org/, to be held October 12-16 in Atlanta, GA.

GAMES FOCUS

Although certain people may believe applications, applications and applications are the key to world domination, some of us know the true key—the reason we became interested in computers—is games. While much of our karmic lineage may come from the punch-card generation of computer hackers, for whom so many games may not have been available, many of us grew up in the days of early home machines such as the Spectrum, Commodore, Atari and Amiga. Some younger hackers may even hail from the days of 3-D. (Imagine having grown up with *that* technology.) Despite slow processors, limited colors, small memories and other obstacles of primitive technology, the years saw numerous ingenious masterpieces, elegant studies in working within limitations, classic games to which we returned time and again, sometimes poking (on BASIC machines) or manipulating with hex editors and disassemblers, but mostly just playing.

Now Linux is developing its own classic games, in a time when limitations on processor speed, color graphics, sound, multitasking, memory, disk space, and networking hardly even seem to exist. Like the classics of old, Linux games have a character all their own; since the games are often developed by a single person or a small group of people, they tend to have a personal, hand-made quality which is missing from their slick, commercial counterparts on other platforms.

Rather than technological limitations, the main constraint today seems to be development time. Although open-source cooperation solves a large part of this problem, another part of the Linux answer is playability, the mysterious quality possessed by games of old which captured our attentions and imaginations despite 1MHz processors and graphics that weren't even vector-shaded 3-D. One legitimate Linux classic which exemplifies this essence of playability, and is an excellent game for inaugurating this new gaming section, is Jan Hubicka's Koules, found at www.paru.cas.cz/~hubicka/koules/English/koules.html.

Koules for X version 1.4

The Dark Applepolisher, you see, is up to no good—he has sent his spherical forces to conquer Earth and claim its resources. In order to defend us from these evil Koules, you will have to bump them out of each of the 100 sectors (and finally confront the Dark Applepolisher). Fortunately, you have been transformed into a yellow beach ball. Well, according to Hubicka, mutated into a chest with eyes to make your job easier, of course.

Although the task may seem simple at first, there will come many varieties of Koules, each with different weights and sizes and mysterious abilities. Black holes and stars and other natural dangers appear as well, and the Koules keep coming! Once the first few sectors are cleared, special Koule ablities and secret weapons begin to appear and the game becomes more exciting and visually interesting. Fortunately, you too can gain special abilities from Koule deserters who give you more weight, more speed and even extra lives. A well-stocked beach ball can weigh enough to wipe out even enormous Koules in a single blow.

Koules is simply an excellent idea which, when developed, becomes fantastically playable. Koules supports up to five players, at a single terminal or over a network, and is more fun with multiple players. Keyboard, mouse and joystick are supported, as well as SVGA and X. Sound support is excellent and exists on multiple platforms, and there are multiple difficulty levels.

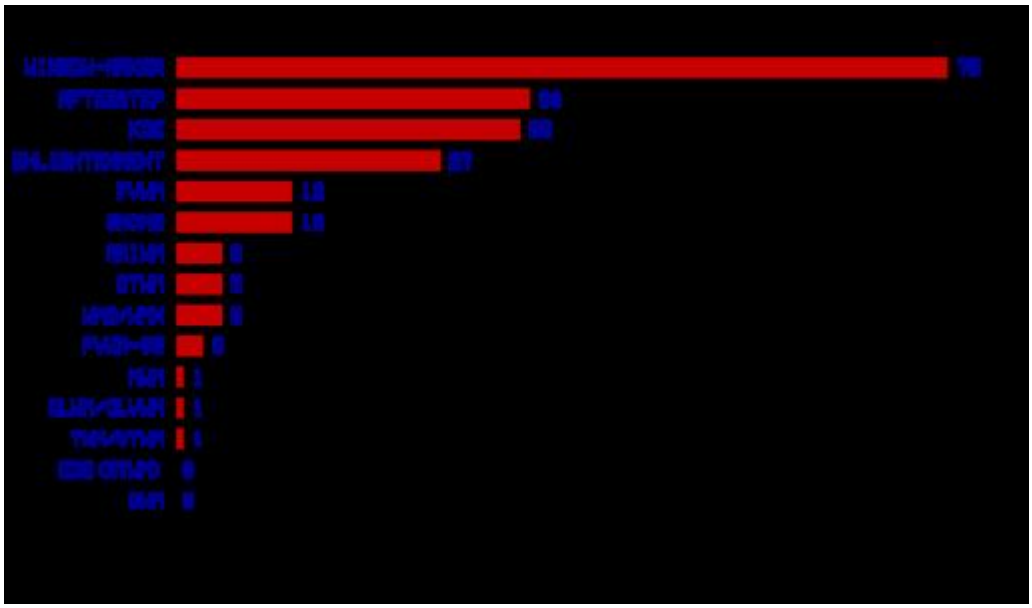
Even though Koules was developed by a single person and is completely free, it has the finished quality of a professional game along with the personality of a small project. I recommend getting some friends together on a rainy Saturday, defending Earth from and finally defeating the Dark Applepolisher, and going out for pizza to celebrate the victory. When you get home, you can start again!

Also, check out the Linux Game Tome at http://happypenguin.org/.

—Jason Kroll

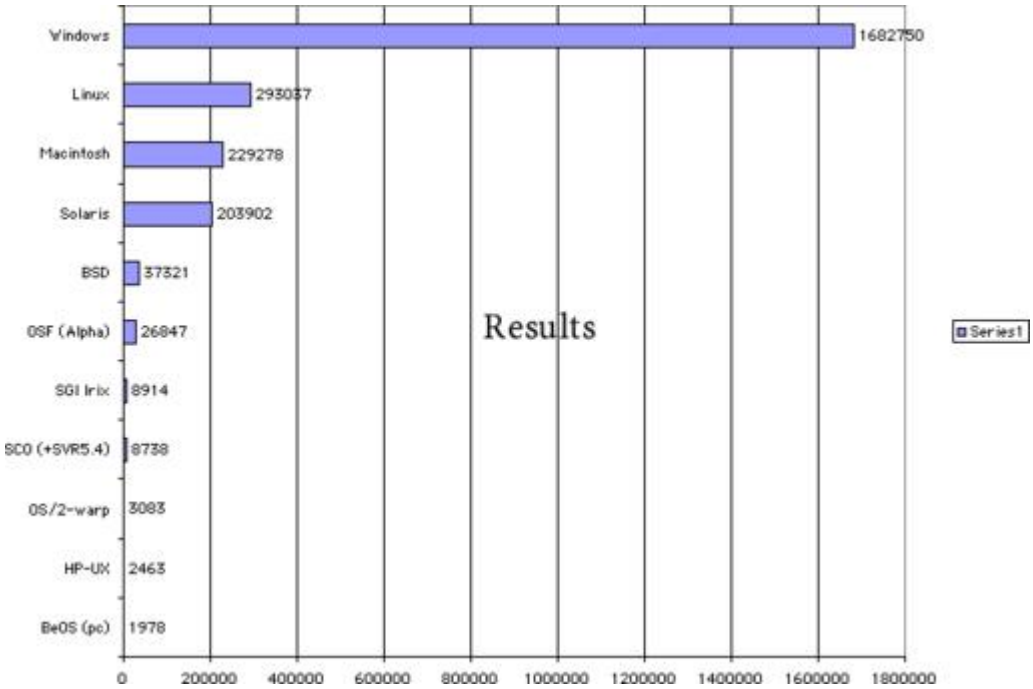
MOST POPULAR WINDOW MANAGER

Matt Chapman's web pages at http://www.plig.org/xwinman/are devoted to Window Managers and offer the browser a chance to vote for his favorite. Here are the results as of June 22.



ET PHONE TUX

What kind of computers are doing the most listening for extraterrestrial intelligence when they're not busy grinding cycles on earthbound work? The first answer is obvious. The second isn't (unless you're one of us, of course). The following graph shows the stats from http://setiathome.ssl.berkeley.edu/as of the most recent summer solstice.



Thinkful Wishing

No-partition Windows-based Linux Installs: Seems like everybody's trying to take the fun out of installing Linux. They want to make it easy. They want to give away the ending and spare us the story. They want to make the hacker's OS as hack-free as possible.

Well, maybe they have a point. And if they do, why not go one better? Why not make Linux installable on any Windows box as it stands? Click on your download file and install the sucker right there, from Windows, *without* partitioning the hard drive.

Sound crazy? Not to Cameron Cooper and Keith Broere, the founders of Phat Linux. These guys have figured out a way to load a full Linux distribution from inside Windows. When it's over, you've got a two-OS box.

The punchline? This isn't new to either of these guys. They've been on the case since they were both 14 years old—last year. Next fall they'll be sophomores in high school. But not the same high school. In fact, not even the same country. Cameron lives in Winnipeg, Manitoba and Keith lives in Sandusky, Ohio (“near the amusement park”). Two guys, two countries, one cool new distribution. Check it out at http://www.phatlinux.com/.

Internet Floppies Revisited: No sooner thought than done. Right on the heels of last month's Internet Floppy idea, the guys at FreeDiskSpace.com have mounted a set of sites that works for all the major platforms, including Linux.

Want extra disk storage—a place to store up to 25MB of files you can pull down from any browser anywhere? Check out http://www.FreeLinuxSpace.com/. Signup is a breeze.

The next step is to make this a value-add for ISPs and anybody else with the space and a way to make money with it. But wait a minute. They do that, too, with an affiliates program, banners and sponsors for the folders in your FreeLinuxSpace directory.

Now, how long will we have to wait for file I/O over the Net? Another whole month?

—Bruce Fryer

Earth-shaking Harbingers

It was kind of amusing, really, fielding brickbats from testosterone-pumped twenty-somethings for whom money and Microsoft's survival are so central that they have trouble grokking that anyone can truly think outside that box. On some subjects their brains just shut down—the style reminded me a lot of the anonymous cowards on Slashdot.

—Eric Raymond to Norm Jacobowitz, June 22, discussing Eric's Microsoft speech (Complete interview can be found at linuxresources.com/articles/linux_review/19990623.html.)

You'd Understand if you Majored in Pheesoox

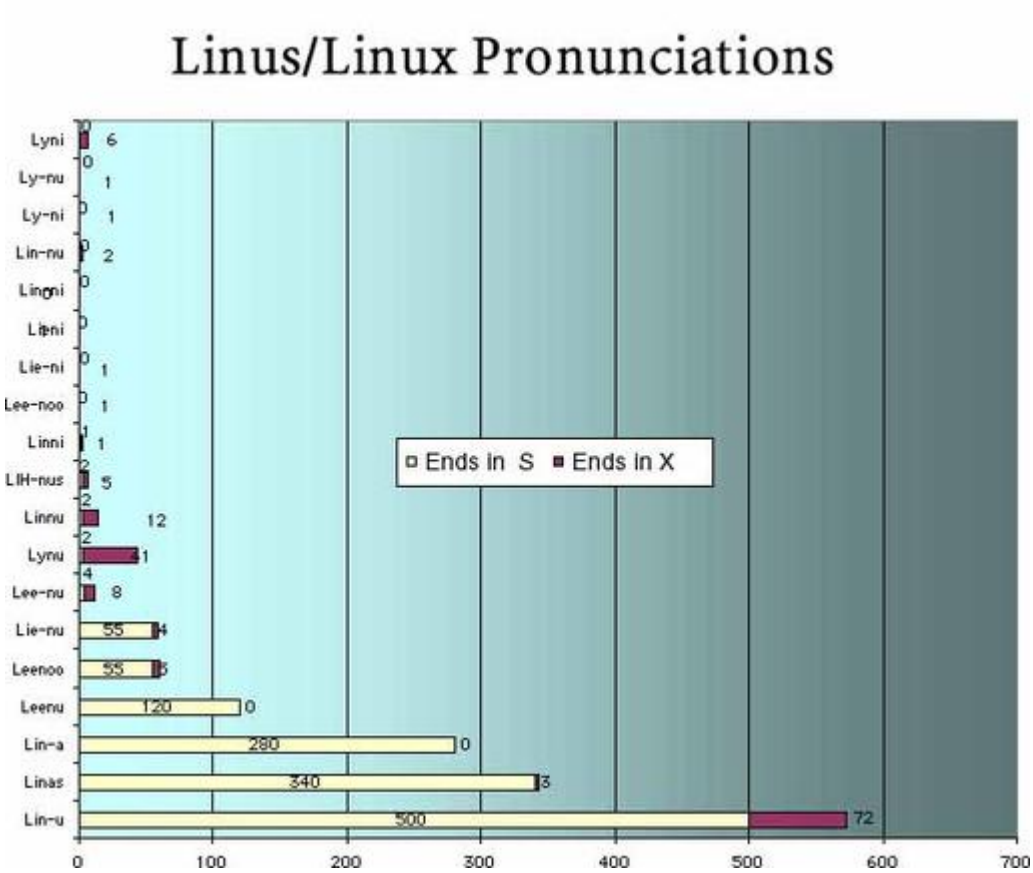
At PC Forum in 1997, Jim Barksdale, then the President and CEO of Netscape, said he got the idea for opening his company's browser source code from “this guy Raymond”, and identified the originator of Linux as “Linus Pauling”. He wasn't too far off, because Linus Torvalds' parents actually named their boy after the famous American Nobel prize winner and vitamin C wacko.

And there began the tale of two pronunciations that have done nothing but permute. There is not only no consensus on how to pronounce Linux, but this condition appears to derive from an equal uncertainty about how to pronounce Linus. The Web is full of sound files in which Linus says, “Hi, my name is Leenoos Torvahlds and I pronounce Leenooks as Leenooks.” More or less. That's Jim Choi's phoneticization of the recording.

But let's face it: that's not complicated enough. Since there is only a one-letter difference between Linus and Linux, we thought we'd see how well the two mapped across the Web by searching for the coincidence of Linus and Linux with various phonetic spellings of the same. As you can see, the results are

equally absorbing and inconclusive, providing plenty of grist for the disagreement mill.

Oh, by the way, Lin-ux appears to be the winner, with 72 pages showing this pronunciation.



Linus/Linux Pronunciations

VENDOR NEWS

The Linux Professional Institute LPI (http://www.lpi.org/), an industry-wide group developing a professional certification program for Linux, is pleased to announce the creation of its corporate sponsorship program and a number of early sponsors. Several new members have been added to its Advisory Council, including IBM, ExecuTrain and CompUSA. Two sponsorship plans, one for corporations and one for individuals, have been introduced to allow anyone to assist the LPI in its goal of creating a high-quality, vendor-neutral program. The LPI aims to deliver its first certification exams in July 1999.

Ecrix Corporation(http://www.vxatape.com/) announced a key partnership with **Penguin Computing Inc.**(http://www.penguincomputing.com/), a company focused exclusively on turn-key Linux solutions. Penguin will offer Ecrix's VXA-1 tape drive on all of its Linux servers, providing a new data backup and restore option for its customers. The VXA-1 features a SCSI-2 interface and storage

capacity of 66GB and transfers data at 6MB/second. VXA tape cartridges are available in two capacities: the V17 stores 66GB and the V6 stores 24GB.

Cygnus Solutions announced the availability of Sourceware CD, a subscription program for the open-source software projects hosted by Cygnus, at http://sourceware.cygnus.com/. The Sourceware CD provides convenient access to the latest open-source technologies, such as eCos (Embedded Cygnus Operating System), the EGCS compiler, GDB debugger and Cygwin.

Pacific HiTech(http://www.turbolinux.com/) announced it has officially changed its name to **TurboLinux, Inc.** This change in corporate identity marks the next milestone in the company's ramp-up of its North American operations after announcements in May of partnerships with IBM and Computer Associates. TurboLinux is a global player in the Linux industry with offices in the U.S., Japan, China and Australia. Its product is currently the fastest-growing operating system platform in Japan. When TurboLinux 3.0 was introduced in Asia in December, it outsold Windows NT (2000) at Japanese retail point-of-sale outlets, according to the technology analyst firm Computer News.

Venture capital firm Kleiner Perkins Caufield and Byers has invested in **Linuxcare**(http://www.linuxcare.com/), a San Francisco-based provider of technical support for Linux. Kleiner Perkins' general partner Ted Schlein is now a member of Linuxcare's board of directors. As part of the investment deal, Fernand Sarrat, former chief executive of Cylink, is Linuxcare's new Chief Executive. Arthur Tyde, Linuxcare's founder, will become Executive Vice President.

SuSE Linux 6.1 is now available at Best Buy, Borders, CompUSA, Fry's, Hastings, Micro Center and Waldenbooks—over 1700 locations nationwide. SuSE Linux can be found in the Operating Systems section of software retailers and in the Computer Books and Software sections of bookstores. SuSE Linux 6.1 features the 2.2.5 kernel and a comprehensive set of applications for home, office, technical and business users.

SuSE GmbH(http://www.suse.com/), the parent company of SuSE Inc., began offering a Business Partner Program targeted specifically at Linux system integrators and consultants. This program is in addition to the recently announced VAR and ISV Partner Programs launched at Spring Comdex '99 by SuSE Inc. The Business Partner Program includes priority support, training, a moderated private on-line forum, and access to a knowledge base, among other features. Qualified Partners are those who seek to offer Linux services and want to benefit from association with the SuSE brand.

O'Reilly & Associates, **Digital Education Systems**(DigitalEd) and **barnesandnoble.com** have signed an agreement for barnesandnoble.com to resell O'Reilly/DigitalEd web-based courses. As part of the agreement, barnesandnoble.com will be the exclusive on-line bookstore reselling the courses. The O'Reilly/DigitalEd courses, based on O'Reilly's best-selling technical books, provide a self-paced on-line learning experience that takes full advantage of the Web's interactivity. barnesandnoble.com will offer 12 courses in O'Reilly's web technology series, plus *Introduction to the Palm Pilot*.

Tripwire Security Systems(http://www.visualcomputing.com/) opened new offices in Washington DC, Chicago and San Francisco to support rapidly growing sales efforts across the U.S., as well as customers in those regions. Tripwire also announced a distributor agreement with Matsushita Inter-Techno Co. in Japan to create broader market awareness for TSS' Tripwire File Integrity Assessment software. The software can identify corrupted systems and files throughout the network, so the servers or workstations can be taken off-line and repaired quickly, minimizing down time and system administration time.

Stop the Presses

OpenSource Forum, a two-day conference on Linux and other emerging open-source software for IT executives, was held on June 30 and July 1 in Austin, Texas. This event, which I attended, was capably presented by Ziff-Davis.

This was a completely different experience for me than attending shows such as LinuxWorld or Linux Expo. The attendees were dressed casually, but were definitely business and professional people—not the hardcore Linux faithful. These people were there to find alternatives to Windows and determine whether open source was a good fit for their companies. Their minds were open, but not made up.

Keynote speeches by Eric Raymond, Ransom Love and Jon “maddog” Hall were enlightening and gave a good positive start to the proceedings. Eric discussed open-source business models and how to decide if and when to go open or stay closed. Ransom talked about the shift from mainframes to PCs (right-shifting) and the current shift to Internet devices (left-shifting), noting Linux is the perfect Internet device because of its capability to be pared down to a very small footprint, its stability, easy customization, high performance and low cost to implement and maintain. Jon discussed the various ways to make money with Linux and advised companies to “put an ad in *Linux Journal*”.

User Friendly Cartoon

Other talks presented a different side. In particular, Jonathan Eunice, President of Illuminata, proclaimed that for large enterprise applications, Linux was

definitely *not* "enterprise-ready" and "free, open-source software is *not* a panacea". He pointed out that the market demands a standard for something it can depend on, that UNIX failed because of fracturing due to not being able to agree on open standards, and that when time is of the essence and skills are limited, paying for a commercial product is the way to go.

Z-D's theme for the show was "Build Your Business with Open Source" and the auditorium was decorated as a construction site. Flashing yellow lights onstage proved to be a bit distracting. Attendance seemed low compared to the Expos and could be numbered in the hundreds rather than the thousands, although I did not get any final count. Still, it was a good conference—one that provided a much-needed platform for Linux and Open Source to strut their stuff for the business world.

—Marjorie Richardson

Readers' Choice Awards

It's that time of year again—time to vote for your favorite products in our Readers' Choice Awards. Voting will be held from September 1 through October 15 on the *Linux Journal* web site, www.linuxjournal.com/. Help your favorite products receive the fame and adulation they deserve—visit the site and fill out the entry form. In the immortal words of James Hoffa, "Vote early and vote often." Winners will be announced in our January 2000 issue.

Rumor Mill: Though neither camp would substantiate the rumor, word has it Adobe Systems, Inc. has shown interest in purchasing Corel Corporation. We're sure Adobe would love to hear your opinions on this one.

Factoid: How do penguins sleep? Some species return to their burrows on land for a few hours of rest, but most penguins take only short naps. Some penguins actually sleep at sea, although this has not yet been observed. Overall, they sleep very little—much like programmers!

Another Famous Linus: Linus Van Pelt: better known simply as "Linus". Famous *Peanuts* character in the long-running strip by Charles Schultz. Noted for trademark "security blanket" and thumb-sucking. Turns 47 on September 19th. Words to live by: "I love mankind. It's people I can't stand."

Rumor Mill: James Sasser, the U.S. Ambassador to China, has blamed much of the tension between the two countries on the recent proliferation of “1999: Year of the Penguin” T-shirts. Graphics and T-shirt designer Jesse Judd was unavailable for comment, having retreated to the Olympic Mountains outside Seattle.

Archive Index Issue Table of Contents

Advanced search

The New Building Trade

Doc Searls

Issue #65, September 1999

Doc on building.

My friend Frank Saelua is a builder. Back in the old country—in his case, Samoa—Frank taught math. I have no idea how much he knows about math, but I do know he is fully capable of building anything or fixing any building. To put it another way, Frank hacks buildings.

Frank was the foreman of the crew who built our house. This was no ordinary construction job. The architect was a 78-year-old, who had been an apprentice of Frank Lloyd Wright and had many of the Great Man's qualities, including the prickly perversity behind such lines as “It's the job of the architect to bankrupt the builder.” While the house was a remodeling job, it was also completely original, turning a one-story ranch into a two-story modern, with cantilevered decks, whole walls of custom-made glass and almost nothing found in a catalog or at Home Depot.

Of course, mistakes were made, as they always are, and minds were changed. Much of the kitchen had to be redone. Pipes in the wrong places had to be moved. A bedroom wall bulged strangely and had to be flattened.

What amazed me was that Frank could look at all these problems—walls, windows, pipes and floors—as if they were modeling clay. As if nothing was a permanent structure. As if making or altering a building were merely a matter of tools and time. Need a door moved? Sure; stay out of the way and we'll do it this afternoon. Sorry about the dust.

I didn't fully understand the similarity between hacking Linux and hacking buildings until our own chief hacker and publisher, Phil Hughes, stayed at our place over the past few days, fixing just about everything that didn't work. This included my home-brew FM transmitter, which had baffled me for nearly a year (and I'm not stupid—except next to guys like Phil and our readers). Armed with

a schematic, a meter and a soldering iron, he fixed the thing in less time than it took him to tell me how bad my tools were and what I should do to replace them. Not much different than his constant put-downs of all editing tools that don't measure up to **vi**.

Where Phil did his best work was on our Linux box, a no-name 133MHz PC clone. Using vi and other software tools, Phil turned it into a mean, clean Linux machine. When he was done, this 133MHz clone was routing e-mail, serving web pages, hosting files for an office full of Macs and PCs, and serving as a desktop for reading and writing .doc, .xls and .ppt files. In other words, he made it into a real Linux system. I submit that this is less an example of Linux doing good work than of a good worker using Linux and UNIX-grade tools to do what lesser materials and tools won't allow.

Phil looked at our Macs the way Frank loooked at the home tool-boxes in our garage, each filled with amateur-grade implements from Sears and Orchard Supply. He made me realize I know even less about building real computing solutions (in the literal sense of that hackneyed word) than I do about building houses. I also realized only those who truly know the virtues of vi and other “Real Tools” are in a position not just to solve problems, but to build a better world.

For proof, look at the Internet. Much of what we know and love about the Internet—such as the way it moves mail and serves up pages—was built by guys who love to solve hard problems with good tools. These are guys who look at computing problems the way Frank Saelua looks at a bad wall.

Of course, there is far more to the Internet than Sendmail and Apache. But I submit there is something highly significant about the success of those solutions—two applications truly deserving of the label—that isn't highly obvious, and that is the matter of origins. There is something about where those problem solvers came from which gave their solutions an enormous scope.

Where they came from was the UNIX world. Back in 1994, when I got my first working account with an ISP, I had a hard time getting my head around all the things my stupid old Macintosh could suddenly do all at once: browsing in multiple windows; archie, gopher and TELNET sessions; file transfers and even web service, thanks to Chuck Shotten's freshly hacked WebStar. One day, I was on the phone with one of the geeks who built the ISP when he interrupted me and said, “You gotta understand: this is UNIX. You can do lots of stuff at once. In fact, there's just about no limit to what you can do.” At the time, his whole business was built on cheap, used Sun machines and a pile of free software.

There's just about no limit to what you can do. Combine the scope of UNIX with a problem-solver's mentality and you've got a future that's equally promising and hard to see from the non-UNIX perspective. Trying to anticipate that future with non-UNIX concepts is like trying to frame a skyscraper with nothing but two-by-fours and sheet rock. And this is what Microsoft is up against right now. Whatever its ambitions, Microsoft will always come from the desktop. From the client. From one person working alone with a *personal* computer.

The future of computing won't be built by a company, even though we'll call it an industry. It will be built by builders and companies of builders. Both will operate on UNIX-informed concepts of not just operating systems and development models, but of understanding and solving problems and—critically, because this is a first—of doing business.

This new building trade won't be limited by one vendor's monopolistic insistence that everything be built only with its materials and tools. It won't be made out of one vendor's pre-fab parts. Most of all, it won't be built on shaky foundations that nobody can improve because their bricks and mortar can be touched only by their own manufacturer.

At too many companies today, even the best builders are limited by software and tools they can get only from the aisles of Microsoft Depot. This will end. Now the software business will turn into a *real* building trade, because the whole conversation will be liberated from hegemonistic corporate agendas by the builders themselves. Where they'll come from is the UNIX world view—one where *there's just about no limit to what you can do*.

This new trade is about designing, assembling, reassembling and fixing structures that are good because good professionals using good tools and materials are doing the work and learning constantly from each other about how to do it better—and doing it for the love of the work far more than the money.

Enabling this community is fundamental to everything we do at *Linux Journal*. Many of our writers are just readers who step forward because they have useful stories to tell. Calling them freelancers doesn't cover their value. Eric Kidd puts it this way in a post at Scripting.com: “For those of you who aren't familiar with *Linux Journal*, it is one of the best geek magazines currently available. In an age when *Dr. Dobb's* and *Byte* have utterly forsaken their technical roots, *LJ* still publishes actual source code—in some issues, well over half the articles have sidebars with program fragments. The articles are written by members of the Linux community.”

So there it is. You made us what we are, and for that, we owe you a hearty thanks. Now let's get back to work.

Doc Searls (info@linuxjournal.com) is a Senior Editor at *Linux Journal*. He also gives us marketing advice we take quite seriously. He telecomputes from Northern California—nice job if you can get it.

Archive Index Issue Table of Contents

Advanced search

New Products

Ellen Dahl

Issue #65, September 1999

AccuRev, SQL Anywhere Studio, Pogo Linux Systems and more.

AccuRev

Ede Development Enterprises, Inc. announced AccuRev, the only software configuration management tool that automatically tracks all aspects of software code changes without imposing any strict development rules or requiring any additional resources, special knowledge or effort. AccuRev's transaction-based database and hot backups guard organizations against data loss due to hardware, network and power failures. AccuRev is available on Linux (Intel and Power PC) and on several other platforms. The cost for a single-user license is $749 US and includes one year of support and updates.

Contact: Ede Development Enterprises, Inc., 350 Haverhill Street, North Reading, MA 01864, 800-383-8170, 978-276-3443 (fax), info@ede.com, http://www.ede.com/.

SQL Anywhere Studio

Sybase, Inc. announced the industry's first mobile and embedded database for Linux, Sybase SQL Anywhere Studio. Sybase's Linux version leverages Linux's enterprise-class reliability and performance on low-cost hardware, such as work group servers, monitoring systems, edge servers or point-of-sale devices. SQL Anywhere Studio enables users to synchronize data seamlessly from enterprise servers and work group servers to laptops and hand-held computing and embedded devices. It is available for $399 US (one user) or $999 US (five users), and supports Red Hat Linux 5.1 and 5.2.

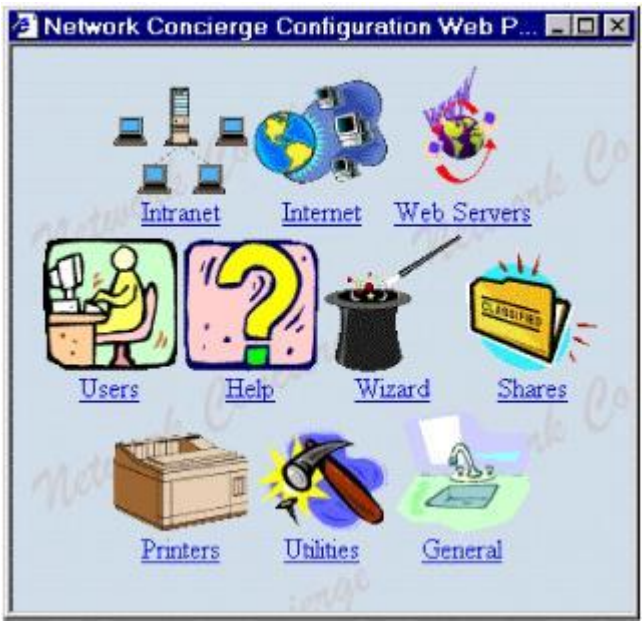
Contact: Sybase, Inc., 6475 Christie Ave., Emeryville, CA 94608, 510-922-3555, http://www.sybase.com/.

Pogo Linux Systems

BuyPogo.Com now offers the reliability and power of the Linux operating system at an affordable price. In July, the company began offering the $299 Pogo and the $599 Pogo Pro through its web site. BuyPogo.Com is targeting the needs of consumers and small business owners who desire the reliability and value of Linux.

Contact: Agenda Technology Group, BuyPogo.com, 555 Bryant St., Palo Alto, CA 94303, 888-828-POGO, http://www.buypogo.com/.

Thin-Server Appliance Software



Network Concierge Inc. introduced the industry's first build-it-yourself Linux thin-server (appliance) software for the small/medium SOHO, school and home network markets. The software provides an easy-to-install, easy-to-use, easy-to-administer browser-based graphical user interface, wizard and on-line installation guide. It can be installed on a spare/new, custom, standard or rack-mountable PC server to build a dedicated, single or multi-function, thin-server

appliance that can be configured as a gateway, e-mail, file, print, web, proxy server, etc. Price is $99 US per network server license.

Contact: Network Concierge Inc., PO Box 1429, Framingham, MA 01701, 877-876-1169, sales@nc4u.com, http://www.nc4u.com/.

Code Fusion

Cygnus Solutions unveiled Cygnus Code Fusion for Linux, the industry's highest performance, most complete Integrated Development Environment (IDE) for Linux developers. The Code Fusion IDE is optimized for the Intel2 architecture and integrates the C, C++ and Java3 programming languages with a robust graphical user interface. Built on top of GNUPro, Cygnus Code Fusion supports all major Linux distributions. It is priced at $299 US.

Contact: Cygnus Solutions, 1325 Chesapeake Terrace, Sunnyvale, CA 94089, 408-542-9600, 408-542-9699 (fax), info@cygnus.com, http://www.cygnus.com/linux/.

HOOPS/AFC

Spatial Inc. and Tech Soft America (TSA) announced the availability of ACIS-enabled HOOPS/AFC from TSA. HOOPS/AFC is a commercial-grade application development framework which enables software developers to create high-performance 3-D modeling applications for Linux and other platforms. The framework of HOOPS integrates ACIS with several user interface toolkits, ensuring optimal design and rendering performance of applications built on these technologies. Please contact Spatial Technology for pricing.

Contact: Spatial Technology Inc., 2425 55th Street, Suite 100, Boulder, CO 80301-5704, 303-544-2900, 303-544-3000 (fax), info@spatial.com, http://www.spatial.com/.

Multiprotocol Routers

ImageStream Internet Solutions unveiled four new Linux-based multiprotocol routers that deliver high performance, high port density and high availability at prices up to 40% less than competing routers. The new ImageStream routers include the single-slot R1, the four-slot Rebel Router, the twelve-slot Gateway Router and the eighteen-slot Enterprise Router. ImageStream routers integrate WAN products from SDL Communications, Inc., which provide support for 56/64K DDS, fractional and full T1/E1, as well as fractional and full T3/E3 network connections. Please contact ImageStream for product pricing.

Contact: ImageStream Internet Solutions, 7900 East 8th Road, Plymouth, IN 46563, 800-813-5123, 219-935-8488 (fax), sales@imagestream-is.com, http://www.imagestream-is.com/.

iASP

Halcyon Software introduced the Instant ASP (iASP) which lets developers deploy Active Server Pages (ASP) or JavaServer Pages (JSP) on all leading Java-enabled operating system platforms, web servers and application servers. A free developer version of iASP, which offers limited concurrent sessions, is available at Halcyon's web site. The price of the standard edition is $495 US. As an open-deployment framework, iASP gives developers true cross-platform deployment capabilities on leading server and operating system platforms, including Linux.

Contact: Halcyon Software, 50 W. San Fernando St. #1015, San Jose, CA 95113, 408-998-1998, 408-998-1922 (fax), sales@halcyonsoft.com, http://www.halcyonsoftware.com/.

LinkScan 5.4 Workstation and Server

Electronic Software Publishing Corporation (Elsop) introduced a new version of LinkScan, 5.4, which includes a new low-cost Workstation version and a graphical user interface for windowing systems. It operates on all UNIX servers including Linux. LinkScan requires web server software and Perl 5. Free fully functional evaluation copies of LinkScan 5.4 can be downloaded from the company's web site. LinkScan Workstation is a single-user implementation designed for individual developers in large enterprises and organizations having smaller web sites of up to 300 documents. A LinkScan Workstation license is $300 US.

LinkScan Server is the complete multi-user, enterprise LinkScan implementation and includes LinkScan/Dispatch. It can handle large web sites comprising 250,000 HTML documents and/or 100,000 external links. A LinkScan 5.4 Server license is $750 US.

Contact: Electronic Software Publishing Corporation, 209-391-9446 (fax), linkscan@elsop.com, http://www.elsop.com/.

XML Pro v2.0

Vervet Logic announced the release of XML Pro v2.0, the next version of the popular extensible Markup Language editor. The upgrade is available via the Vervet web site. Registered users of XML Pro can upgrade from v1.2 free of charge. Priced at $174.95 US for the CD-ROM and $149.95 US for download, XML Pro v2.0 offers many new features including full W3C XML 1.0 compliance, integration of the IBM XML4J parser, drag and drop, undo, ability to change document encoding, View DTD and Java 2 (JDK 1.2.1) support. The XML Pro/Near & Far Designer bundle is available for $299 US from both Vervet Logic and Microstar. Supported platforms include Linux.

Contact: Vervet Logic, 501 North Morton, Suite 211, Bloomington, IN 47404, 812-856-5270, 812-855-4506 (fax), sales@vervet.com, http://www.vervet.com/.

RS2000 Remote Access Card

Ariel Corporation announced its PCI-based RS2000 remote access card for PCs running Linux. Together, Linux and the RS2000 provide a scalable, low-cost, high-availability platform for adding high-density V.34, 56K and Basic Rate ISDN remote access to enterprise systems and ISP points of presence. It combines dual T1/PRI interfaces with 24 V.90 modems on a single PCI plug-in card. The RS2000 is available immediately from Ariel for a list price of $6995 US.

Ariel also announced a new driver development kit (DDK) that makes it easy for OEMs to integrate Ariel's high-density PCI and CompactPCI remote access cards with the operating system of their choice. The DDK is part of Ariel's new OEM Starter Kit, which includes the DDK, an RS2000 or RS2000C card, Windows NT, Linux and a five-year warranty with spare-in-the-air support. Pricing starts at $8795 US.

Contact: Ariel Corporation, 2540 Route 130, Cranbury, NJ 08512, 609-860-2900, 609-860-1155 (fax), info@ariel.com, http://www.ariel.com/.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #65, September 1999

Our experts answer your technical questions.

Adding New Users

I am having extreme difficulty adding users on Red Hat 5.1. I've used various methods to add them to my system via the **adduser** command and through various X control-panel programs. When I add them, everything seems to go well, and they exist when I look them up after I created them. My problem is that I can't log in without being root. If I attempt to log in as a user, it will say “login incorrect”, as if I typed in the wrong password or user name, which is not the case. I have tried numerous times and am frustrated since I cannot get any users to work on my system. —Anthony Dipaula, dipaula@udel.edu

You will need to check two files. The first is /etc/passwd. Make sure each user has a valid shell and home directory. Then look at /etc/login.conf. This defines login control settings, and you might have a setting that prevents non-root logins on the console. —Chad Robinson, chadr@brt.com

File Type

Can someone tell me what the hyphen after a file name indicates? Two examples are in /etc: **passwd-** and **group-**. Thanks in advance for the answer. —George R. Boyko, grb99@nni.com

These are backups made by the utilities that manage these files. You can most likely delete them safely, but it's also a good idea to keep them around. Otherwise, if your passwd file ever becomes corrupted, you will be unable to log in, and rebuilding it is always a pain if you don't have a good copy lying around. —Chad Robinson, chadr@brt.com

Voice/Faxmodem Problem

I have a multi-platform computer running Windows 95, Windows NT 4.0 and SuSE Linux (kernel 2.0.36). My problem is that I don't know how to install the Plug and Play modem on Linux. With the first two operating systems (Windows 95 and NT), the modem works just fine. I tried many ways of installing the modem using the Isa PNP tools. It recognizes the modem, but I still have trouble setting up the port. I read some of the HOWTOs and they don't help me much. They say to set up the modem with the **setserial** command, but I don't know how to use this command. In the Windows NT 4.0 setup, my modem is installed with the following parameters: COM5, IRQ 07, Input/Output Range 02E8 to 02EF.

Can someone tell me how to set up my modem on Linux? Or how to set up the modem using the **setserial** command? It is a 3Com US Robotics 56K Voice Faxmodem. —Manuel Enrique, phrotos@email.com

COM5 is usually (not always) a clue that your modem isn't a modem but a winmodem, which would therefore be useless with anything other than Windows.

Now, if it really is a modem, you should find out which port it is on as far as Linux is concerned (probably ttyS2) with **dmesg | grep ttyS** right after boot. Then configure the interrupt and the port number like this:

```
setserial /dev/ttyS2 port 0x3E8 irq 2
```

substituting the right values for your card. —Marc Merlin, marc@merlins.org

Java Crashing Netscape?

I am somewhat more than a newbie, but when it comes to things I don't know, I am clueless. I am running on Red Hat 6.0, and Netscape Communicator 4.5. Whenever I go to a Java web site, like linux.com or yahoo.com, Netscape will crash; that is, it will just disappear. If I run Netscape in a terminal ./netscape, it will also disappear whenever I go to a Java site. But in the terminal, it will say “bus error”. What does this mean? I know I enabled Java in my setting, but how can I fix this? —Eric Zabinski, diablo@elnet.com

Netscape has been known to crash for a variety of reasons, many of them linked to Javascript and especially Java. I recommend you try upgrading to Netscape 4.61, or downgrading to Netscape 4.08. —Marc Merlin, marc@merlins.org

PPP and Network Configuration

I recently accomplished my first successful Linux install, Red Hat 5.2. I installed PPP software, but thought network configuration should be done only for NIC-equipped machines, not ones just doing dialup, so I didn't perform the network configuration at that time.

Although I have subsequently edited a raft of PPP-related files using the HOWTOs, books and the suggestions of an ISP, I don't get the expected PPP "garbage" (40-character frames with frequent { characters) in response to a **pppd** command from the command-line prompt. Does this clearly mean that I don't have PPP support properly installed and need to reinstall or add via a package manager? Or would this also occur with errors in the configuration files (like host.deny or host.allow or ...)? —Stephen S. Rinsler, 70353.714@compuserve.com

You should be able to create the PPP connection with **netcfg**, which you can launch from the control panel that should be there by default when you log in as root and launch X (type **control-panel** otherwise). You also have the option of using **linuxconf** to create your PPP connection. —Marc Merlin, marc@merlins.org

Mounting NFS

I need to implement NFS (Network File System) from my Linux host to a WinNT Server for files access. I have Linux Red Hat 5.2 installed on a Pentium 133 MHz and 24MB RAM machine. I also installed the NFS services (client and server). In a server machine, I have Windows NT 4.0. I read the book *TCP/IP Illustrated Vol. 1* by Richard Stevens (chapter 29 talks about NFS), so I am a beginner of the protocol. When I try to mount an NFS link, the following error appears:

```
mount: RPC: Port mapper failure - RPC: Unable to send
```

Can you help me? —Ing. Juan Salazar Velasco, jsalazar@merkafon.com

The easiest way to do this is not with NFS, but with Samba, which you can get at samba.anu.edu.au. By installing Samba on your Linux box, you will be able to use **smbclient** to access your Windows NT server, and the combination of **nmbd** and **smbd** to allow your NT server to access files on your Linux box. This works for printers, too.

NFS services require special configuration in the Windows NT server (and usually additional software), and often aren't as fast because the native protocols to each type of server (SMB for NT and NFS for UNIX) were designed

with somewhat different intentions in mind. In my experience, UNIX emulates SMB better than NT emulates NFS. —Chad Robinson, chadr@brt.com

It's not clear to me which machine is the NFS server. If it's Linux, then most likely the portmapper isn't running. Type:

```
/etc/rc.d/init.d/portmap restart
```

Also, make sure that rpc.nfsd and rpc.mountd are running on your machine.

If the server is NT, type:

```
showmount -e name_of_nt_server
```

and as long as you don't get suitable output from it, your NT server isn't configured correctly. —Marc Merlin, marc@merlins.org

Hard Drive Problems

While trying to install Linux 5.2 from the *Linux For Dummies* book, I have somehow locked myself and the install program out of it. Once you have partitioned a drive partially for Linux, how do you get back in and straighten out any mess you may have made? The install floppy or the CD-ROM have nothing on them that I can access from the DOS prompt or from the DOS program **fdisk**, which, by the way, says there is an error reading the disk and won't let me in. Also, DiskDruid won't let me back in either. Can you help? —"Budskie", Budskie@email.msn.com

I would guess that you somehow damaged your partition table, but this is tough to tell without looking further at the drive. You may wish to try a different tool, such as the **fdisk** program that comes with the Slackware package. It is a more raw tool, and while it may be harder to use, it probably won't completely stop you from getting to your drive.

I've seen problems like this come from misunderstandings about the LBA (logical block addressing) setting in the BIOS for a drive. Unfortunately, without knowing more about what exactly has happened, I can tell you only how to completely wipe out what you've done so you can reload your system. (The adage about backups comes to mind here.) If you do go this route, you would want to use a more basic tool (such as the fdisk that ships with Slackware) to remove all of the Linux partitions on your drive in the hopes of recovering your DOS information. Failing that, you could always remove them all and re-install Windows to return to a stable state, then try again. —Chad Robinson, chadr@brt.com

Archive Index Issue Table of Contents

Advanced search

Adventure

Joseph Pranevich

Issue #65, September 1999

A trip down gaming's memory lane with an enthusiastic, long-time player.

Adventure is an old game and one that has been known by many names: ''Adventure'', “Colossal Cave”, and the simplistic abbreviation “Advent”. As games go, it is indeed a classic, having sparked an entirely new tangent of game development in the early days of the gaming industry. Its effects can still be felt today.

I sat down this morning to write a review of a game. At some point, I figured I'd tell about its strengths and weaknesses. In the end, I'd probably assign it a value in stars or thumbs or joysticks or some other arbitrary measurement of quality. However, assigning a letter of quality to this game is nearly as anachronistic as seeing *The Illiad* on Oprah's booklist; it is simply inappropriate and demeans the true quality of the work. So instead, I'm going to tell you what this game has meant to me, a little about where it came from, and how it shaped the Quake III world in which we live today. We've come a long way from the early days of text adventures—or have we?

My Story

It was 1989 and I was 10. The end of the eighties was upon us, and we all looked with trepidation to the coming decade and the closing years of our millennium. The Bangles played “Walk Like an Egyptian” to hordes of onlookers not interested in the nostalgia of the moment. The economy was in a deep recession, and with the Cold War and movies like *War Games*, people were naturally wary of what the coming years would bring. The Berlin Wall fell that year, and we all wondered what more the future would bring. Well, I didn't wonder. I was 10, what do you expect?

Instead, I was concentrating on my own life. Already the budding geek (before the term became a compliment), I went out and hunted my butterflies (tilted

my windmills) and enjoyed life. I had a Commodore 64 at the time, and thought it was the coolest thing in the world, but it never affected me. It was fun, certainly. I knew how to program simple things in Basic. I could make balloons float around on the screen. I knew that if I typed “SYS 64738”, the whole thing would reboot. But it never grew beyond a tinker toy to me—it never awakened my inner geek.

That changed, of course. My father bought me a game package he had pulled from the bargain bin. It was a set of two 5.25-inch floppies from Broderbund (the covered wagon people) entitled “Golden Oldies Volume 1.” Loading it up, I was surprised to find a collection of four “classics” of the day: Adventure, Eliza, Life and Pong. My mother decided she liked Eliza the best and would continually attempt to convince the poor thing that she was a beach ball or something similar. I found my passion in Adventure, a little game that transported me into a world filled with scary little dwarves and myriad treasures, all for the taking, if the young spelunker was up to the challenge. Needless to say, I never beat the game. I rarely drew maps, and when I did, I routinely forgot which direction was east. I had managed to memorize a large portion of the game, but in retrospect I don't think I ever truly got it. Maybe it was because I was too busy experiencing the world to really want to win. (Or more likely, because I was young and stupid, but leave me to rose-tint my world however I want, please.) Eventually, my old copy of Adventure developed bad sectors and I put it to rest.

Several times later in life, I would rediscover these games I loved. I did eventually get to play Zork (a version of Adventure) for the Apple II, and found it richly gratifying, but too similar to the game I left behind. Other games I loved: King's Quest, Quest for Glory and the early Sierra masterpieces. Somehow, even with their fancy four-color pictures and their beeps, they never transported me to a world in the same way my first Adventure did. I rediscovered Adventure when I finally turned to the “dark side” and was given a 386 and a modem to play with. It was just as I remembered it, except free from a local BBS. I played and played and mapped it a little, but I still never beat it. Life had a tendency to intrude and I put it away again, a cherished childhood game to relive later.

I was happy to rediscover my game in the BSD Gamespack (/usr/games/) that ships with many Linux distributions, including Red Hat. It was like meeting a childhood friend again on the street. Now I'd like to introduce it to you.

History of Adventure

The game of Adventure has a long history, dating back to the earliest days of modern computing. In the beginning (1972), there was Will Crowther. He was a programmer, caver and role-player (Dungeons and Dragons, in particular).

Faced with boredom (the motivational force behind many a good program), a desire to create something for his daughter and his assorted talents, he set out to create a game based on his explorations of the Mammoth and Flint Ridge cave systems in Kentucky. This game included many of the features of the later versions and included descriptions and room names taken directly from the caves in the *real* world.

A couple of years later, Don Woods discovered the game and added a number of fantasy elements to the plot including pesky dwarves, a dragon and whatnot. His primary inspiration was fantasy literature (such as the *Lord of the Rings* trilogy), and Adventure was never to be the same again.

Many other coders added bits and pieces after that point. This may be one of the earliest examples of open-source gaming. The game Adventure could be one of a dozen or more variants, each with minor scoring differences and occasional major additions. By and large, all Adventures are created equal.

An Evolutionary "Dead End"?

Most people, while writing game reviews, don't have the luxury of jumping ahead 30 years and seeing how a particular piece of software affected the computing landscape. I do have this luxury.

The games of today, at first glance, are not even remotely similar to the games of yesteryear. Space Invaders yielded to Quake and Rogue to Diablo. Are these games fundamentally different from their legacy counterparts? Take a close look and you will find relatively little difference, except for the quality of the graphics engine. Early in the process, game manufacturers discovered that graphics sell. In the early days, black and white games yielded to those with color and then yielded to those with sound. 3-D polygonal rendering is the big kick these days (take a look at Nintendo 64). Are these true enhancements, or just eye candy? I tend to take the stance of the latter. Move, shoot and move, shoot and repeat until your thumbs get sore. What game am I talking about?

It would be foolish of me to point to two of the oldest breakthroughs in game design and say that the gaming industry has stagnated—that is not what I'm trying to say. Now-classics such as Civilization, SimCity, PacMan, Tetris, The Legend of Zelda and Super Mario Brothers sparked their legions of clones but were all individual and profound. But which of these series haven't suffered in some way from graphics mania? (I'd like to make an exception to this point. Mario 64 was actually, in my opinion, a breakthrough in gaming and not just a knock-off of a superior product.)

What games trace the lineage back to Adventure? What games descended from its pinnacle in the non-combat adventure genre?

Following Adventure first and foremost was Zork. There were probably others before Zork, but Zork made a noticeable impact on our consciousness. The original Zork didn't stray far from the original concept; it had a newer and fancier parser than the old caving adventure, resulting in more fluid game play. Later installments by the Infocom crowd took gaming to new heights. Their games had magic, mystery and fun. They pioneered the interactive story and took it much farther than even Adventure. This is not to say that each of their games were classic; many were derivative, but many more were unique.

Eventually, however, Infocom's luster began to wear off. Games increasingly turned to graphical albeit inferior forms of gaming. Sierra On-line took a prominent role with King's Quest and other games. Their storytelling was fantastic, but something was just not right about using the arrow keys to walk up to a tree and typing “look into tree” when you got there. Gradually, they refined that interface further with icons and mouse movement, but the “universal range of motion” feel that you got with the old text adventures was gone. Now your actions were defined by what you could click on, not by what you could imagine. Particularly frustrating for me were the layers of eye candy they added to their games. It looked nice, but was often distracting, especially when you wanted to manipulate and just couldn't figure out how.

Eventually, “point and click” adventure gaming became even more “clicky” with the advent of games without funny icons. In these games, you just click on the screen and the computer figures out what action you want from context. To make matters worse, it wasn't long before gamers discovered a profound secret: when one is stuck, one needs only to click madly around the screen until stumbling on the magic hot-spot that jumps to the next level. Certainly, this is a far cry from the seemingly infinite worlds offered by the early text-adventure games, but is it progress?

Infocom (now owned by Activision) and others would try to break away from this mold with later games, such as Return to Zork, which offered a much more detailed mouse interface. Sadly, however, this is where I became disinterested with the genre of non-combat “adventure” gaming, and I have yet to see whether their more recent titles have matched that level in playability. (I'd love to see a review of Zork: Grand Inquisitor—too bad they don't make it for Linux.)

What made these games so wonderful? It wasn't the graphics, obviously. What made these games special was something more subtle. In addition to generally good writing, the textual format allowed game designers to plug the computer's output directly into the gamer's imagination. Who among us doesn't have splendid memories of “Flood Control Dam III” (Zork), the little white house with the boarded front door (Zork), or the “Hall of the Mountain King” (Adventure)?

Graphical games just don't plug into the subconscious in the same way. For me, that made all the difference.

The Game

In the game of Adventure, you are an unnamed hero-explorer with a good head on your shoulders and a sturdy back. Your mission is to locate and explore Colossal Cave and bring back the hoards of treasure rumored to be inside. But watch out; magic is afoot in the cave and all may not be as it seems. Scoring is based on how many treasures you find, how much of the cave (if any) you explore, and how many of those treasures you get out of the cave.

The game play is fairly simple. You instruct the game in two-word English phrases (“get lamp”) to do things, and your character does them. The vocabulary of the game is not quite as large as its Infocom descendants, but that is understandable. When you get stuck, you can ask for help (although it may not be forthcoming) and when you die, you can get put back together again. It is all very simple.

On first starting the game, it will probably seem confusing. You've just been plopped down in the middle of a large world with little idea of where to go. A building is nearby—that's obvious. Consider it your base of operations, as you'll need to deliver your hoards of treasure there in order to score the most points. Farther afield is the hidden entrance to the cave, a confusing forest (the game's first mini-maze), and many other sights once you get underground. That is, if you remembered to bring your lamp. Otherwise, it'll be a short trip.

If you're playing this game for the first time, I recommend just exploring for a while and seeing the sights. Once you are comfortable with the interface, you'll probably want to get pen and paper and start drawing maps like a true spelunker. Without some sort of map, you will most likely become lost in the maze-like passageways, halls and crawls of Colossal Cave. If you do get stuck, don't ask me for help. I'm at Witt's 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.

Archive Index Issue Table of Contents

Advanced search

Remotely Monitoring a Satellite Instrument

Guy Beaver

Issue #65, September 1999

How a small aerospace company uses Linux to remotely monitor the performance of a satellite instrument.

G & A Technical Software (GATS) is a small aerospace company located in Newport News, Virginia. Our primary area of business is the analytical support of satellite-based atmospheric remote-sensing projects. We started using Linux in 1995 for software development and data processing workstations.

NASA is funding a two-year satellite mission called TIMED (Thermosphere Ionosphere Mesosphere Energetics and Dynamics) to study the atmosphere. One of the experiments on the spacecraft is called SABER, which stands for Sounding of the Atmosphere using Broadband Emission Radiometry. SABER's mission is to make measurements of temperature, ozone, carbon dioxide, water vapor and other trace gases to learn more about the complex relation of energy transfer between the upper and lower atmosphere. GATS has been contracted by NASA to develop and operate the systems and software that process the data from the spacecraft.

A big part of the SABER experiment is calibrating and testing the instrument while it is on the ground. This task involves making measurements of known sources and analyzing the results so that data taken from orbit can be understood. Calibration of SABER is a difficult task, because measurements need to be made at conditions found in space. To accommodate this, SABER is calibrated in a large chamber that is capable of cold temperatures and near vacuum pressures. SABER and its calibration facility were built by the Space Dynamics Laboratory (SDL) in Logan, Utah. Figure 1 is a photograph of the actual calibration test happening in the testing facility at SDL.

Figure 1. The high bay at SDL. The SABER test chamber is under a clean tent on the left. To the center and right are engineers operating the instrument during calibration tests. The computers mentioned in this article are along the right wall.

In this article, I will describe a Linux-based system that allows remote monitoring of the SABER calibration tests. I will discuss the porting of software from a Windows NT workstation to a Linux workstation due to reliability requirements, as well as the use of several open-source products such as the GNU C++ compiler, CVS for configuration management, Xmgr for diagnostic plots, PostgreSQL database, VNC for remote terminal access and Perl. Many of the systems proven by this project can be used by other small businesses for powerful cross-country data processing while keeping total costs and development time low. The system is robust and provides real-time monitoring and analysis of the SABER instrument (located in Logan, Utah) from Newport News, Virginia. The same system will be used for post-launch data processing.

Requirements

Big projects go through several iterations of requirements/design/review, and SABER was no exception. When the time came to put together a final design, the requirements for the system were well-documented. Basically, the system needed to talk via a socket to a computer (a Sun workstation dubbed GSE for ground system equipment). The GSE computer provided raw data from the SABER instrument as well as all the test equipment connected to it. We needed to unpack the data into large staging files which could be read by analysis programs. On top of this, the system needed to access a PostgreSQL database on the GSE that was being populated by the SDL test operators with information on the various tests being run, such as times and readings from the various sensors. We had to provide software flexible enough to quickly analyze data with unknown quirks. This means easy debugging and code-level

diagnostic plotting capability. The system had to work 24 hours per day and provide remote access from Virginia to Utah so that our engineers could have timely access to the data for supporting the actual tests. Finally, the system we developed had to be easily reconfigurable for post-launch data processing.

Design

We began the design by using software that GATS developed for another project. These C routines provide an interface to raw spacecraft data in the standard spacecraft format developed by the Consultative Committee for Space Data Systems (CCSDS). SDL simplified the job for us by making the raw CCSDS data available over a TCP socket, which our system already supported. We call this stage of the processing Level 0. It takes the raw binary data in packet form and unpacks it into ASCII files, sorted by data type. We chose ASCII for the initial design because disk space was not an issue (9GB total), and troubleshooting is much simpler if files are human-readable. ASCII is also a friendly format for passing between different systems. We wanted this stage of processing to write across the network as well, so NFS was a must. Note that utilizing socket calls and NFS at this stage means the physical location of the Level 0 computer is irrelevant, as long as the Internet is available.

The next stage of the processing is called Level 1, and its first job is to query the PostgreSQL database for test information. With this, the relevant data is extracted from the Level 0 files and broken out into a single file for each test. For this part of the design, Linux was a must. The Level 1 processing had to talk to an NT box and a Sun workstation. We chose Linux because Samba is easy to set up for NT file sharing, and data sharing with the Sun is automatic via NFS. The software design called for object-oriented programming (OOP) using C++, so that classes writing the Level 1 files could be reused downstream to read the data. Linux comes with GNU C++, which is extremely reliable and easy to debug. Another reason for choosing Linux was that the analysis tools for this project were developed in C++ on a Linux workstation using the powerful (and freely available) display package called Xmgr.

This design is easily modified for post-launch processing, since the data coming from the calibration setup is in precisely the same format as that coming from the spacecraft.

Implementation

Our original design was based on two computers. One was an NT workstation running the Level 0 software and simultaneously providing some real-time strip charts of data. The second was a Linux workstation running the Level 1 software. It was a P-II with 128MB RAM and 24GB hard drive to catch six week's worth of calibration data. We built the Linux workstation for about $3500. We

ran Samba on the Linux workstation so that the NT box could write its Level 0 files to the large hard drive on the Linux box across the network.

We flew out to Utah with the computers to test the original design. It worked, but we had some reliability issues with the NT-based Level 0 software. One requirement was that the Level 0 files must be generated reliably and be read-accessible 24 hours per day. We had some unexplained hiccups when the files were accessed by programs such as Microsoft Notepad. After verifying that file read/write access was set correctly, we decided to change the design and port the code to the Linux box.

As mentioned above, the Level 0 software was reconfigured from another GATS project that required an NT workstation. After reviewing the code, we realized the only Windows unique code was in the winsock calls for socket connections. It is easy to port Windows **<winsock.h>** calls to GNU C **<socket.h>** calls. In fact, it entails deleting some overhead needed in Windows but not required in GNU C. To make the code backward-compatible, we simply wrapped the Linux code with **#ifdef LINUX-#else** pre-compiler directives. This allowed us to keep the same version of code that worked on NT and Linux under one configuration management version. Some examples of converting Windows socket calls to Linux are shown in Listing 1.

Listing 1.

With these modifications, we now had the Level 0 and Level 1 processing stages on one Linux workstation. We called this the Calibration Analysis Computer, and we left it and the (now spare) NT workstation at SDL connected to their network. During calibration tests, it generated Level 0 files 24 hours per day and never had to be rebooted over the six-week calibration period. As I mentioned before, the NT workstation had some strip-charting capabilities for viewing real-time data. This turned out to be a good use for the NT box, so we configured it to work with SABER data. Since the Linux workstation was on the Internet, we automatically had remote access, and we needed the same for the NT box. VNC filled the bill. This remarkable application (Virtual Network Computer) pipes the Windows desktop to a client running on a Linux X session. With VNC, we had the ability to set up and monitor the NT box remotely so we could configure it for SDL engineers wishing to view real-time temperature output. We could also view the same real-time strip-charts on our Linux workstations back in Virginia.

This system offers a great deal of flexibility. We chose to let the Linux workstation at SDL connect to the socket, then access the data through the Internet. We could also connect to the socket from Virginia and generate the Level 0 files in our office.

The Level 1 processing stage ran flawlessly. The PostgreSQL database on the GSE workstation was easily accessible from the front-end library (libpq-fe.h) that comes with this powerful SQL database. Each calibration test event was performed automatically by a script on the GSE workstation which automatically populated an “event” in the database. The Level 1 stage made a query to this database for beginning and ending times of the test event. With this information, the particular piece of the Level 0 files could be pulled out and processed (even though they were constantly being written to). These files, called calibration analysis files, could then be accessed by the analysis routines, which we called Level 1b.

The Level 1b processing stage contained powerful tools for analyzing the calibration data. Many of the algorithms were from other GATS projects and were reconfigured to be methods within classes developed for the SABER project. One valuable diagnostic tool proved to be the C-callable library that came with the Xmgr graphics analysis package. These library calls were wrapped in an easily utilized plotting method contained in our Level 1b class. Using objects that have diagnostic plot methods shortened the debugging period that comes with looking at real data for the first time.

Our development team was lean and mean—three people working on various modules with support from two others, all working under the CVS configuration management system. Since our computers moved back and forth across the country, they were set up to be easily configured for their current location. We did this with simple scripts, stored in /root, which are run after bootup. We had a script for each location—“atGats” and “atSDL”. These simple scripts set the local IP address and set an /etc/resolv.conf (containing the location of local name servers and IP addresses) for each location. The scripts simply made a dynamic link to the appropriate resolv.conf file. An alternate solution would have been dynamically assigned IP addresses through DHCP, but we already had pre-assigned local addresses from SDL, and this method was simple and gave us easy control based on the computer's location.

Listing 2.

I attended the first two weeks of calibration testing in Utah to ensure everything was working well, while my support people stayed home in Virginia. During that time, we easily diagnosed problems and were able to make updates to the code using CVS (Concurrent Version System, the GNU configuration management package). I described problems, they were fixed in Virginia, and I got instant updates with the CVS update command. This works because CVS can be set up with an NFS-mounted CVS root directory on a remote machine (in this case, at GATS in Virginia).

Once the testing started and Level 0 files were being generated, we monitored the data from Virginia as the tests were being run. Quick queries of the database with SQL told us when tests were completed. Making the Level 1 files was easy to automate at this point. Since the Level 1 software had command-line arguments typical of UNIX applications, we wrote Perl scripts to loop over the test event IDs (which were database fields designating each test event), and generated the Level 1 files in batches. As we migrate the software to the post-launch processing system, we will automate the entire daily processing with similar Perl scripts.

Results/Benefits

This system provides us with the ability to remotely monitor the SABER calibration tests and the flexibility to remotely process the data, or even download and process it in Virginia on our local Linux workstations. We've used Linux for five years, so we were not surprised at the ease of getting Linux to work in a networked environment. From this exercise, we verified the validity and robustness of Linux as the OS of choice for systems requiring remote access and remote monitoring. Invaluable to our project has been the flexibility Linux offers for configuring a mobile computer to operate in different locations under different networks. Finally, the ease of troubleshooting provided by the open-source software available in the standard Linux packages makes it the clear winner in rapid-development environments.

Resources

Acknowledgments

Guy Beaver is a Software Engineer and Senior Associate at GATS Incorporated. He has worked with computers and satellite remote sensing since 1984. Although he looks young, he is old enough to remember card punches and 8-track tapes. He can be reached at beaver@gats-inc.com.

Archive Index Issue Table of Contents

Advanced search

First UNIX/Linux National Competition Held in Ljubljana, Slovenia

Primoz Peterlin

Ales Kosir

Issue #65, September 1999

A look at the questions and answers for a contest to find Linux solutions to common problems.

On April 24, 1999, the Slovenian National Linux User Group organized the first national competition in UNIX/Linux. The competition was among the activities connected with the 23rd Annual National Competition in Computer Science for high school students and the Fifth Festival of Computing, both of which were held in late April at the Department of Computer Science at the University of Ljubljana. As opposed to the already long-established tradition of national competitions in computer science, where a great emphasis is given to problems easily solved with procedural programming languages, this competition seeks its niche in favoring solutions implemented with “alternative” tools from the UNIX programming environment, where the strengths of these tools can be shown in the best way.

When preparing the exercises for this competition, those of us who served as the organizing committee searched the Web extensively for similar competitions elsewhere. Unfortunately, we found none. We would therefore like to make our own experiences available to the broader audience and hopefully initiate some collaboration in this area. In this article, we present the exercises that were part of the competition together with their solutions, explained in detail and commented. We conclude with the results of the competition accompanied by a few remarks and afterthoughts.

Exercises

2:1. Frequency Analysis of Text

Perform a simple frequency analysis of text. Read a text file and write to the standard output the complete list of all words in the text along with their frequencies. The list should be sorted in ascending order, with the least frequent words on the top and the most frequent ones on the bottom. A "frequency" is a number which tells us how many times a certain word occurs in the given text. You can assume the text file contains no other characters but letters and blanks.

2:2. vi Editor

On a multi-user system, you want to prevent multiple users from editing the same file simultaneously using the **vi** editor. Suggest a solution! Implement your solution as a script. Comment the good and the bad aspects of your solution. You can assume that every user calls the editor using the command **vi** *filename*. You cannot rely on the **vi** editor issuing a warning when editing an already-opened file.

2:3. Shuffle

The lines in a given file are sorted by certain criterion. You don't want this, and would like to shuffle the lines pseudo-randomly. Propose your shuffling algorithm and implement it in the code. Pay attention to the efficiency of your code. "Pseudo-randomly" means you are allowed to use the random number generator available in the tool you will choose.

2:4. IP Addresses

A text file contains multiple references to IP addresses. You want to replace them all with the fully qualified domain names (FQDN). The IP addresses and the corresponding FQDNs are listed in the /etc/hosts file:

```
193.2.1.72      nanos.arnes.si
```

You can assume this file contains nothing but records like the one shown.

Numerical IP value is of the form 0.0.0.0-255.255.255.255. Without sacrificing much generality, you can assume every pattern 0.0.0.0-999.999.999.999 in the given text file is a valid IP address and its corresponding FQDN can be found in the /etc/hosts file. You can also assume each IP address in the text file is delimited by at least one blank character on both sides.

General Rules

You are allowed to use the script languages of any command shell (e.g., **sh**, **csh**, **ksh**, **bash**) or any script language such as **sed**, **awk** and **perl**. You are also

allowed to use all the user commands provided by a UNIX system as specified by POSIX. You are *not* allowed to use compiled languages such as C, C++, Pascal or FORTRAN. If you are in doubt as to whether the tool you plan to use is allowed, you can ask a member of the supervising committee. The decision of the supervising committee is final.

Solutions

2:1. Frequency Analysis of Text

It is easy to solve this exercise with the **sort** and **uniq** commands and some command pipelines. The solution can be written in a single line:

```
tr " " "\n" < filename | sort | uniq -c | sort -n
```

With **tr** we replace each blank in the text file with a newline, thus chopping it into a form where a single word is written per line. Since tr expects the data stream from the standard input, we have to redirect the data from the file to the standard input with the < sign. The author of the exercise has made life a bit easier for us, since we don't have to worry about the periods, commas and other non-letters in the text. The output from tr is pipelined (the | sign) to the input of the sort command, which outputs an alphabetically sorted list of all words in the text file. We pipeline it again to the input of the uniq command. With no options, the uniq command eliminates multiple occurrences of adjacent lines in a text. With the option **-c** (count), it also reports the number of occurrences for each line read on the input. Since we have already chopped the text beforehand, uniq thus outputs the list of frequencies. The only remaining task is sorting it by frequency rather than alphabetical order, which we do with the second **sort** command. The **-n** option is used to perform the sort by the numerical value of the first field.

Was the explanation too quick? If you are not familiar with pipelines, we advise you to try and assemble the complete operation one step at a time. To start, use the tr command alone:

```
tr " " "\n" < filename
```

Now try to pipeline its output into sort, and watch what happens! Continue with the rest of the pipelined commands.

Let us conclude this discussion with an illustration showing frequency analysis on an actual text. The ten most common words in *Martin Krpan*, a text by Fran Levstik, are:

```
61 v
65 Krpan
74 bi
```

```
74 na
107 da
121 in
127 ne
142 se
161 pa
207 je
```

2:2. vi Editor

This exercise requires familiarity with the concept of file locking. The given program (in our case, the **vi** editor) is renamed or moved to a directory not included in the users' **PATH** environment variable. In its place, we put an enveloping script named vi which, when called, creates a lock file, starts the editor and removes the lock file on exit. The lock file is created in the same directory as the file we are editing, and not in the /tmp directory, for instance. This is necessary to avoid any confusion which could arise when two files with the same name exist in two different directories.

A possible problem may occur if two users simultaneously start editing a certain file. User A starts the script, which discovers a lock file does not exist and proceeds to create one. Before it actually creates it—remember, UNIX is a multitasking system—the process is interrupted by user B starting the same script. Since the lock file still doesn't exist, the script also decides that user B is allowed to set the lock.

The problem can be avoided if the procedure is reversed: first we create the lock file, and afterwards we check if we were successful. Does that sound strange? The solution presented here does exactly that. Using the **touch** command, the script always tries to create a lock file, then reacts depending on the exit status reported by touch. On a successful exit, touch exits with a zero (true) exit code, while on an unsuccessful exit, it reports a non-zero (false) exit code. There are other possible situations in which touch exits with a non-zero exit code, but here we are focusing on only one. If we try to touch a file belonging to another user for which we don't have write permission, touch reports an error and exits with a non-zero exit code. Therefore, we have to make sure that when the lock file is created, no other user has write permission. This is accomplished using the **umask** command as shown below: user (**u**) is the only one who is granted both read and write (**rw**) permissions, while users from the same group (**g**) and all the other (**o**) users are granted none. Since we don't want the umask command to affect our environment, we have limited its influence by enclosing it in parentheses. The gibberish following the touch command is used to send the error messages produced by touch into oblivion. We are interested only in the exit code and are happy with touch running absolutely quietly.

```
#!/bin/ksh
if ( umask u=rw,g=,o=; touch $1.lock >\
/dev/null 2>&1 )
```

```
  then
    vi $1
    rm -f $1.lock
  else
    echo "$1 is currently locked."
  fi
```

Since it depends on the touch exit code, the solution presented here doesn't prevent a user from opening *his own* file more than once. This is not explicitly required in the exercise, so it is not considered a deficiency in this context. There is also nothing preventing *root* from opening anybody's files. Moreover, the solution employing an enveloping script does not prevent any user from reading an already-opened file from *within* the vi editor. Unlike the traditional AT&T vi, many of its newer incarnations have the file-locking facility built in. **vim** (Vi IMproved), the most common vi replacement on Linux, is one of them.

2:3. Shuffle

Shuffling is obviously a procedure that can easily be accomplished by brute force. First, each line is prepended by a random number, then the lines in the file are sorted according to these random numbers, and finally all the prepended numbers are cut away, retaining only the original lines in their altered order. Again, the solution can be written in a single line:

```
  awk '{ print rand(), $0 }' sorted.lst | sort -n
  |
  cut -d " " -f 2-
```

The first command performs the following statement in the **awk** scripting language on the file sorted.lst:

```
  { print rand(), $0 }
```

For those not familiar with awk, commands in this scripting language have the form

```
  condition { statement }
```

For each record on input (if not specified otherwise, a record defaults to a line, as in the present case), the condition is checked, and if met, the corresponding statement is executed.

In our case, no condition was specified, which means the statement is executed for each line on input. The statement itself says to print a random number followed by the line read on the input. The output from awk is then piped to the input of sort, which sorts the lines by the numerical value of their first field. The first field is the random number which was prepended in the previous step.

Finally, we remove the prepended random numbers with **cut**. The two options given to **cut** mean, treat blank as field delimiter (**-d " "**) and extract the fields

from the second field onward (**-f 2-**). We have thus dropped the first field, which was the prepended random number.

Sorting makes the above algorithm an *N* log *N* one. A more efficient shuffling was invented by Durstenfeld (*Comm. ACM* **7**, 1964, 420), having a linear dependency. Here we present an implementation in Perl for our purpose:

```
#!/usr/bin/perl
@line =<>; # the complete input file
                    #is read into a vector
for ($i = 0; $i <= $#line; $i++) { # for i-th line
    $rnd = $i + int(rand($#line - $i + 1));
                    # we select a random line
                    # between i and the end
    print $line [$rnd]; # print it
    $line [$rnd] = $line[$i]; # replace with i-th
}
```

2:4. IP addresses

This exercise requires recognizing the following pattern in the text: a blank, followed by one, two or at most three digits, a dot, again one, two or three digits, another dot, again one, two or three digits, yet another dot, once more one, two or three digits, and the final blank. This pattern can be efficiently described as a *regular expression* in Perl:

```
\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b
```

Here **\b** denotes a word boundary, **\.** is an explicit dot (otherwise a dot means "any character" in the regular expression syntax) and **\d{1,3}** means a sequence from one to three digits.

The solution consists of two separate tasks. First, we have to read the **/etc/hosts** file and construct a mapping table. Second, we scan the text file for IP addresses and replace them with the corresponding FQDN from the mapping table. Again we present a solution in Perl:

```
#!/usr/bin/perl
open (HOSTS_FILE, "/etc/hosts");
while ($line = <HOSTS_FILE>) {
    chomp $line;
    ($ip, $fqdn) = split(/ +/, $line);
    $hostbyip {$ip} = $fqdn;
}
close(HOSTS_FILE);
while (<>) {
  s/\b(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})
  \b/$hostbyip{$1}/g;
  print;
}
```

When constructing the mapping table **$hostbyip**, we used a nice feature of Perl called associative arrays or hashes, which allows us to refer to an element in the table by a symbolic name. Here, we used the IP address written as a string

(variable **$ip**). The address replacement in the second task is implemented using the **s** (substitution) operator: s/*pattern*/*replacement*/g. The final **g** (global) modifier makes the substitution operator more eager; the first pattern found in the line doesn't satisfy it, but it continues to scan the rest of the line for another occurrence(s) of the given pattern.

In reality, the Domain Name System has replaced the /etc/hosts files long ago. For a more realistic example, we would have to replace the reading of /etc/hosts table by a **gethostbyaddr** call. The modified program is actually even shorter:

```
#!/usr/bin/perl -p
BEGIN {
    sub gethostbyip {
    my ( $ip ) =  @_ ;
    $packaddr = pack ("C4", split (/\./, $ip) );
    ($name) = gethostbyaddr( $packaddr, 2 );
    return $name;
    }
}
s/\b(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})
\b/gethostbyip($1)/eg;
```

Results

Thirteen competitors entered this year. They had 90 minutes to solve all exercises using pencil and paper. They were allowed to use all available literature, but did not have access to a computer to test their solutions. Therefore, the committee decided to judge favorably all solutions exhibiting the correct ideas, even if they were not written syntactically error-free. After reviewing all submitted solutions, the committee decided to award prizes to the following three contestants:

- Andraz Tori
- Mitja Bezget
- Gasper Fele-Zorz

The average quality of the submitted solutions suggests that despite all the publicity Linux received during the last year, high school students are not particularly familiar with the tools from the UNIX/Linux programming environment. This is a pity, since we believe that the many strong scripting languages and modular tools are one of Linux's advantages over its competitors.

The analysis of the anonymous questionnaire which the competitors were asked to fill in also yielded some interesting results. The competitors were asked questions about their own computing environment, to classify the exercises on a scale of 1 to 5 from “easy” to “difficult” and to give suggestions. Some found the limit to scripting languages too restrictive. An interesting

response came from a competitor who considered the exercises would be simple “provided that C or C++ could be used”.

This calls for a comment. Every exercise is easiest to solve in a language one is familiar with. Anybody familiar with both C and some scripting language would probably agree that these exercises can be solved in the latter with much less effort. This was intentional. What wasn't intentional is that we discovered the fact that high school students hardly touch on any scripting language at all and are thus unaware of their benefits. Rapid prototyping, for example, is quickly and easily accomplished from readily available tools. This is a complement rather than a replacement for the compiled languages. If speed is truly important, a successful prototype is normally followed by a compiled version, usually distinguished by much faster execution. In practice, both approaches coexist, while in our schools, one seems to have a complete dominance.

Last but not least, the title of this competition also probably deserves a comment. UNIX, or Linux in its narrower sense, denotes the operating system kernel. Kernel-level exercises were not part of this competition and, given the limited time and resources available to competitors, would not be feasible at this moment. It would be honest to admit that in the trade-off between short and catchy names and long and precise ones, we have leaned towards the former. Who knows—perhaps the extra room we have created for ourselves will even prove useful at some later time.

Primoz Peterlin holds a M.Sc. in Physics and works for the Institute of Biophysics, University of Ljubljana, Slovenia. Getting his first computer in 1983, he discovered Linux in 1992 and has never left it since. His diverse interests include biocomputing, typography and mountain hiking. He can be reached at primoz.peterlin@biofiz.mf.uni-lj.si.

Ales Kosir has a BS in Mathematical Physics and an MS in Mechanical Engineering. In the company Hermes SoftLab, he works on compiler-based technology, while he is interested in many aspects of localization. He enjoys cooking, particularly seafood, and for his own pleasure he plays the piano. More about him can be found on nl.ijs.si/GNUsl/.

Filters

Paul Dunne

Issue #65, September 1999

This article is about filtering, a very powerful facility available to every Linux user, but one which migrants from other operating systems may find new and unusual.

At its most basic level, a filter is a program that accepts input, transforms it and outputs the transformed data. The idea of the filter is closely associated with several ideas that are part of the UNIX operating system: standard input and output, input/output redirection and pipes.

Standard input and output refer to default locations from which a program will take input and to which it will write output. The standard input (STDIN) for a program running interactively at the command line is the keyboard; the standard output (STDOUT) is the terminal screen.

With input/output redirection, a program can take input or send output using a location other than standard input or output—a file, for example. Redirection of STDIN is accomplished using the < symbol, redirection of STDOUT by >. For example,

```
ls > list
```

redirects the output of the **ls** command, which would normally go to the screen, into a file called **list**. Similarly,

```
cat < list
```

redirects the input for **cat**, which in the absence of a file name would be expected from the keyboard, to come from the file **list**--so we output the contents of that file to the screen.

Pipes are a means of connecting programs together through I/O redirection. The symbol for pipe is |. For example,

```
ls | less
```

is a common way of comfortably viewing the output from a directory listing where there are more files than will fit on the screen.

Simple programs provided as standard with your Linux system can be enhanced by using them as filters for other similar programs. I'll also show how simple programs of your own can be built to meet custom filtering needs.

One program I don't look at in this article is Perl. Perl is a programming language in its own right, and filters are language-independent.

grep

The program **grep**, “Get Regular Expression and Print”, is a good place to begin. (See “Take Command: grep” by Jan Rooijackers, March 1999.) The principle of grep is quite simple: search the input for a pattern, then output the pattern. For example,

```
grep 'Linus Torvalds' *
```

searches all files in the current directory for Linus' name.

Various command-line switches may be used to modify grep's behaviour. For example, if we aren't sure about case, we can write

```
grep -y 'linus torvalds' *
```

The **-y** switch tells grep to match without considering case. If you use any upper-case letters in the pattern, however, they will still match only upper-case. (This is broken in GNU grep, which simply ignores case when given the **-y** switch —that's what the **-i** switch does).

With just this bit of information about grep, it is easy to construct a practical application. For example, you could store name and address details in a file to create a searchable address book.

Extended Grep

Sometimes, basic grep won't do. For instance, suppose we want to find all occurrences of a text string which could possibly be a reference to Linus. Clearly, searching for **'Linus Torvalds'** is not enough—that won't find just Linus or Torvalds. We need some way of saying, “This or this or this”. Here is where **egrep** (extended grep) comes in. This handy program modifies standard grep to provide just such a conditional syntax by using the | character to denote “or”.

```
egrep 'Linus Torvalds|L\. Torvalds|Mr\. Torvalds' *
```

will now find most ways of naming the inventor of Linux. Note the backslash to "escape" the period. Since it is a special character in regular expressions, we must tell egrep not to interpret it as a "magic" character.

A Note on Regular Expressions

tr

tr is perhaps the epitome of filters. (See "Take Command: A Little Devil Called tr" by Hans de Vreught, September, 1998.) Short for translate, tr basically does what its full name suggests: it changes a given character or set of characters to another character or set of characters. This is done by mapping input characters to output characters. An example will make this clear:

```
tr A-Z a-z
```

changes upper-case letters to lower-case. A-Z is shorthand for "all the letters from A to Z".

sort

Sorting is a very basic computer operation. It is commonly used on text, to get lists in alphabetical order or to sort a numbered list. Linux has a powerful filter for sorting called, logically enough, **sort**.

head and tail

These two very simple filters have a surprising variety of uses. As their names suggest, **head** shows the head of a file, while **tail** shows the end. By default, both show the first or last ten lines respectively, and tail in particular has a number of other useful options. (See the man pages.)

Programmable Filters

Sometimes we need to do something a bit more complex than the relatively simple command lines of the above examples. For this, we need something I'll call a "programmable filter", that is, a filter with a scripting language that allows us to specify complex operations.

sed

sed, the stream editor, is a filter typically used to operate on lines of text as an alternative to using an interactive editor. (See "Take Command: Good Ol' sed" by Hans de Vreught, April 1999.) There are times when firing up vi or Emacs and making the change, whether manually or using vi/ex commands, is not

appropriate. For example, what if you have to make the same changes to fifty files? What if you need to change a string, but are not sure exactly in which files it occurs?

As is common in the UNIX world, where tools are often duplicated in different ways, sed can do most things grep does. Here is a simple grep in sed:

```
sed -n '/Linus Torvalds/p' filename
```

All this does is read standard input and print only those lines containing the string “Linus Torvalds”.

The default with sed is to pass standard input to standard output unchanged. To make it do anything useful, you must give it instructions. In our first example, we searched for the string by enclosing it in forward slashes (//) and told sed to print any line with that string in it with the **p** option. The **-n** option ensured that no other lines would be printed. Remember, the default behaviour is to print everything.

If this were all sed could do, we would be better off sticking with grep. However, sed's forte is as a stream editor, changing text files according to the rules you supply. Let's take a simple example.

```
sed 's/Torvuls/Torvalds/g' filename
```

This uses the sed “substitute” (**s** option) and applies it globally (**g** option). It looks for every occurrence of “Torvuls” and changes it to “Torvalds”. Without the **g** command at the end, it would change only the first occurrence of “Torvuls” on each line.

```
sed '/^From /,/^$/d' filename
```

This searches the standard input for the word “From” at the beginning of a line, followed by a space, and deletes all the lines from the line containing that pattern up to and including the first blank line, which is represented by **^$**, i.e., a beginning of line (^) followed immediately by an end of line ($). In plain English, it strips out the header from a Usenet posting you have saved in a file.

Double-spacing a text file takes just one command:

```
sed G filename > file.doublespaced
```

According to our manual page, all this does is “append the contents of the hold space to the current text buffer”. That is, for each line, we output the contents of a buffer sed uses to store text. Since we haven't put anything in there, it is empty. However, in sed, appending this buffer adds a new line, regardless of

whether there is anything in the buffer. So, the effect is to add an extra new line to each line, thus double-spacing the output.

AWK

Another very useful filter is the AWK programming language. (See “The AWK Tools” by Lou Iacona, May 1999.) Despite the weird name, it is an everyday tool.

To start with, let's look again at yet another way to do a grep: 'grep'. Fast

```
awk '/Linus Torvalds/'
```

Like grep and sed, AWK can search for text patterns. As with sed, each pattern can be associated with an action. If no action is supplied as in the above example, the default is to print each line where the pattern is matched. Alternatively, if no pattern is supplied, then the default action is to apply the action to every line. An AWK script for centering lines in a file is shown in Listing 1.

Listing 1.

AWK's strength is in its ability to treat data as tabular, that is, arranged in rows and columns. Each input line is automatically split into fields. The default field separator is “white space”, i.e., blanks and tabs, but can be changed to any character you want. Many UNIX utilities produce this sort of tabular output. In our next section, we'll see how this tabular format can be sent as input to AWK using a shell construction we haven't seen yet.

Pipes: When One Filter Isn't Enough

The basic principle of the pipe (|) is that it allows us to connect the standard output of one program with the standard input of another. (See “Introduction to Named Pipes” by Andy Vaught, September 1997.) A moment's thought should make the usefulness of this when combined with filters quite obvious. We can build complex instructions 'programs', on the command line or in a shell script, simply by stringing filters together.

The filter **wc** (word count) puts its output in four columns by default. Instead of specifying the **-c** switch to count only characters, give this command:

```
wc lj.filters | awk ' { print $3 } '
```

This takes the output of wc:

```
258    1558    8921 lj.filters
```

and filters it to print only the third column, the character count, to the screen:

```
8921
```

If you want to print the whole input line, use **$0** instead of **$3**.

Another handy filtering pipe is one that does a simple filtering of **ls -a** output in order to see only the hidden files:

```
ls -a| grep ^[.].*
```

Of course, pipes greatly increase the power of programmable filters such as sed and awk.

Data stored in simple ASCII tables can be manipulated by AWK. As a simple example, consider the weights and measures converter shown in Listing 2. We have a simple text file of conversions:

```
From    To      Rate---     ---     ----
kg      lb      2.20
lb      kg      0.4536
st      lb      14
lb      st      0.07
kg      st      0.15
st      kg      6.35
in      cm      2.54
cm      in      0.394
```

To execute the script, give the command:

```
weightconv 100 kg lb
```

The result returned is:

```
220
```

Listing 2.

Power Filters

The classic example of “filtered pipelines” is from the book *The UNIX Programming Environment*:

```
cat $* |tr -sc A-Za-z '\012' |
sort |
uniq -c |
sort -n |
tail
```

First, we concatenate all the input into one file using **cat**. Next, we put each word on a separate line using **tr**: the **-s** squeezes, the **-c** means to use the complement of the pattern given, i.e., anything that's not A-Za-z. Together, they strip out all characters that don't make up words and replace them with a new line; this has the effect of putting each word on a separate line. Then we feed

the output of tr into **uniq**, which strips out duplicates and, with the **-c** argument, prints a count of the number of times a duplicate word has been found. We then sort numerically (**-n**), which gives us a list of words ordered by frequency. Finally, we print only the last ten lines of the output. We now have a simple word frequency counter. For any text input, it will output a list of the ten most frequently used words.

Conclusion

The combination of filters and pipes is very powerful, because it allows you to break down tasks and then pick the best tool for each task. Many jobs that would otherwise have to be handled in a programming language can be done under Linux by stringing together a few simple filters on the command line. Even when a programming language must be used for a particularly complicated filter, you still save a lot of development effort by doing as much as possible using existing tools.

I hope this article has given you some idea of this power. Working with your Linux box should be both easier and more productive using filters and pipes.

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue65/2479.tgz.

Paul Dunne (paul@dunne.ie.eu.org) is an Irish writer and consultant who specializes in Linux. The only deadline he has ever met was the one for his very first article. His home page is at http://www.cix.co.uk/~dunnp/

Archive Index Issue Table of Contents

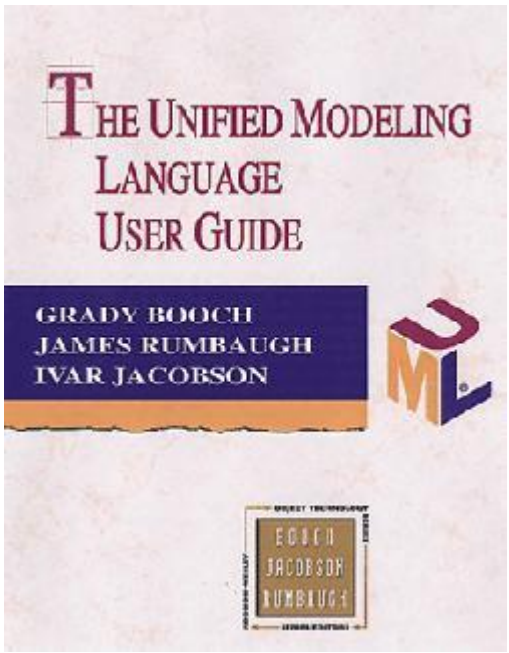
Advanced search

The Unified Modeling Language User Guide

Geoff Glasson

Issue #65, September 1999

This book is intended to be read by people involved in the production of object-oriented software systems.



- Authors: Grady Booch, James Rumbaugh and Ivar Jacobsen
- Publisher: Addison-Wesley Publishing Co.
- URL: www.awl.com
- Price: $47.95
- ISBN: 0-201-57168-4
- Reviewer: Geoff Glasson

The preface of *The Unified Modeling Language User Guide* states:

> The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing and documenting the artifacts of a software-intensive

> system. The UML gives you a standard way to write a system's blueprints, covering conceptual things, such as business processes and system functions, as well as concrete things, such as classes written in a specific programming language, database schemas, and reusable software components.

This book is intended to be read by people involved in the production of object-oriented software systems. It assumes the reader has a basic understanding of object-oriented concepts, and from there teaches one how to use UML. In realistic terms, the book contains too much information to review adequately here, as I cannot do justice to the authors in such a short space.

The book is divided into three major sections: structural modeling, behavioral modeling and architectural modeling. The structural and behavioral modeling are then subdivided into basic and advanced sections. Each section is laid out in a manner such that you can read the entire section (as I did) or only those parts that are important to you.

The basic structural modeling section describes the use of classes, relationships and class diagrams. It provides the basics which all object-oriented software engineers require to build UML models. Classes are described down to their lowest level, and subsections show how to distribute the responsibilities of a system among the classes that compose it. The advanced section expands on this to describe how to model the semantics of a class, object diagrams, packages, relationships and interfaces. This section shows how to model the relationships between classes and how to model the interfaces provided by a group of classes. It also teaches how to model a set of objects and their relationships at a given point in time using object diagrams.

The basic behavioral modeling section describes how to model the interaction between objects. It describes the interaction, use case and activity diagrams which dictate how the objects in the model interact with each other. The advanced behavioral modeling section also describes how to model events, state machines, processes, threads and time constraints. This book contains a wealth of information for the modeling of the structure and behavior of classes and objects. The authors show how to use state diagrams to model the behavior of individual objects within the system, showing how an object responds to internal and external stimuli. The examples start at simple state diagrams and continue through to complex diagrams that contain states and substates (states within states).

The architectural modeling section describes how to use the UML to document the physical arrangement of the software system. It deals with the complex issue of managing the components of the final product as well as how those components are deployed when the software is installed, using component and

deployment diagrams. A component diagram graphically depicts the components that comprise the software; similarly, the deployment diagram describes how the components are distributed across one or more systems and how the hardware components are interconnected.

The authors have gone to great lengths to produce a readable book full of examples and useful tidbits. Try not to be overwhelmed by the volume of information in this book—much information is there, but it is all aimed at helping you model systems more accurately, and thereby design and implement better systems.

In my opinion, this book is a great addition to every object-oriented software developer's library, because it contains a great deal of information about how to model software systems. Although I have used both the Booch method and UML for some time now, I learned many things that improved the quality of my designs, especially in the area of communicating these designs to others. It is a book worth buying.

Geoff Glasson (glastech@iinet.net.au) has been a professional software developer for ten years, working mainly with C and C++ on Solaris systems. He has used UML for 2.5 years. He is currently the leader of a small software development team at Motherwell Information Systems working in the process control field. When he is not working, he enjoys playing with his two kids, spending time with his wife, playing indoor cricket and playing with Linux.

Archive Index Issue Table of Contents

Advanced search