Linux Journal Issue #72/April 2000

Focus

The Internet *by Marjorie Richardson*

Features

The Linux Home Network *by Preston F. Crow*
Everything you need to know to make those Internet and intranet connections at home.

Assessing the Security of Your Web Applications *by Nalneesh Gaur*
This article outlines key test areas to identify security issues in a web application and provide measures to minimize them.

Setting Up a Linux Gateway *by Lawrence Teo*
Setting up a Linux gateway can be a rewarding experience in both home and commercial environments.

Linux and the Next Generation Internet *by Stan McClellan, Michael Stricklen and Bob Cummings*
This article describes the authors' implementation of a demonstration environment for differentiated Internet services (DiffServ) using Linux-based routers.

Eid Eid, OE/ONE Corporation *by Marjorie Richardson*
Internet appliances are the next wave of computers to be bought by the public. Mr. Eid tells us how his company is serving this market.

Forum

Pakistan On-Line *by Rafeeq Ur Rehman*

Linux is being used by an ISP in Pakistan—Mr. Rehman tells us why.

Building Your Own Internet Site *by Tony Dean*

A quick look at what you need to build a web site for your personal or business needs, with pointers to the details.

The (not so) Wonderful World of DSL *by Jason Schumaker*

Discover the joys of using the new technology of DSL and cable modems to make your link to the world.

RTAI: Real Time Application Interface *by P. Mantegazza, E. Bianchi, L. Dozio, S. Papacharalambous,*

An introduction to RTAI for deterministic and preemptive real-time behaviour for Linux.

Novell Adopts OpenLDAP *by Craig Knudsen*

What's new at Novell...

Artists' Guide to the Desktop, Part 2 *by Michael J. Hammel*

In this episode, Mr. Hammel tells us all about Enlightenment 0.16.1.

Transmeta Rewrites the Rules *by Linley Gwennap*

All about the revolutionary new chip from Transmeta.

Reviews

NetMax Apache Webserver *by Allan Liska*

A Practical Guide to SNMPv3 and Network Management *by Charles Curley*

Interconnections, 2nd Ed. *by He Zhu*

Columns

Linux Apprentice: Customizing Vim Some great customizations to Vim's default behavior—make Vim work for you. *by Dan Puckett*

Linux Means Business Converting from SCO Xenix to Linux *by Fred Treasure*

A computer consultant's experience in converting the SBT accounting system to Linux for the Maxwell House Department Stores.

System Administration Large-Scale Linux Configuration Management *by Paul Anderson*

Mr. Anderson describes some general principles and techniques for installing and maintaining configurations on a large number of hosts and describes in detail the local configuration system at Edinburgh University.

Kernel Korner The Linux Scheduler *by Moshe Bar*

A look at how the kernel schedules tasks for both uni-processor and multi-processor machines.

Cooking with Linux Cookie Cutters, Munchers and Crunchers *by Marcel Gagné*

Some clues for protecting your privacy on the Internet.

At the Forge Designing Databases *by Reuven M. Lerner*

Games We Play: Commodore 64 Game Emulation There must have been some magic in that old grey box... *by Jason Kroll*

Focus on Software *by David A. Bandel*

Embedded Systems News Briefs *by Rick Lehrbaum*

The Last Word *by Stan Kelly-Bootle*

Departments

Letters
upFRONT
Penguin's Progress: Where Have the Nets Come From? *by Peter H. Salus*
Linux for Suits : Now What: Are We Going to Let AOL Turn the Net into TV 2.0... *by Doc Searls*
Best of Technical Support
New Products

Strictly On-Line

Web Analysis Using Analog *by Gaelyne R. Gasson*
Every web site needs a way to get accurate statitics—here's a freely available program to give you that information.
Shell Functions and Path Variables, part 2 *by Stephen Collyer*
Mr. Collyer continues his discussion with a detailed description of the addpath function.
The Generation Gap *by Brian Marshall*
This paper examines the issues involved with the use of open-source software components in closed-source applications.
Enlightenment Basics *by Michael J. Hammel*
A guide to getting and installing Enlightenment.

Archive Index

Advanced search

Focus: The Internet

Marjorie Richardson

Issue #72, April 2000

The Internet is taking over our lives—talk about world domination; it has won.

The Internet is taking over our lives—talk about world domination; it has won. Advertisements on TV, billboards, essentially anywhere, all carry the familiar www.dot.com. It has become *de rigueur* for a business to have its own web site. And customers are finding these sites and using them. I buy DVDs, flowers and cards on the Internet; a co-worker buys her groceries there; another does all his gift shopping. The day may come when department and grocery stores are no longer needed, and it might not be very far off.

Statistics show that maybe 30% of U.S. households have computers with access to the Net. This is going to go up rapidly in the next few years as Internet appliances—computers for the computer-illiterate—come to the market. These book-size computers will offer Internet connectivity, and not much more. They will be as easy to use as a VCR. People who never thought about buying a computer before will buy one to find out why everyone is talking about and using the Web. Then they will be hooked too—just like the rest of us.

A company that is betting on this is OE/ONE, and I talked to Mr. Eid Eid about his new start-up and the innovative software they are building for this potentially lucrative appliance market. Mr. Eid is a very personable and forward-thinking man with some interesting ideas about how the Internet will be used in the future.

QoS

Quality of Service support in the kernel has created a new controversy for our community. ISPs now have a way to control traffic so that those who pay more can have priority over those who don't, getting faster connections and faster response times to problems. Linus created Linux and gave it to the community through the Internet. Programmers develop code and share it through the

Internet. Without the Internet, Linux would not exist. The Internet has always been a place where all are equal. There is now the possibility that will not always be true.

—Marjorie Richardson, Editor in Chief

Contents

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The Linux Home Network

Preston F. Crow

Issue #72, April 2000

Everything you need to know to make those Internet and intranet connections at home.

I have a fairly complicated home network that uses many of the different features of Linux. While my network setup is probably more complicated than what you'll ever need in your home, you may find that you want to use some of the same features. In this article, I'll describe how my computers are physically connected, discuss how I want my network to work, and then explain how I used various aspects of Linux to accomplish those goals.

The Physical Layer

I have three desktop computers and a laptop, with Internet access provided by a cable modem and an ISDN line. One of the desktop computers runs Linux and handles communication between the Internet and the internal network.

The cable modem is an external device that converts the cable signal into an Ethernet signal. I also use an external ISDN modem that works the same way. Each of these is connected to a separate Ethernet card in my central Linux system.

The three desktop computers are connected to each other using Ethernet with a hub. The laptop, however, doesn't have Ethernet, so it connects to my central Linux system using a cable between the parallel ports.

So, the central computer has three Ethernet cards: one for the cable modem, one for the ISDN modem, and one for the hub connecting to the other machines.

How It Should Work

First, let me explain how my Internet connections work. The cable modem provides a single dynamic IP address and a fast connection to the general Internet. The ISDN line provided by my company connects to its internal network, and provides a 16 IP address subnet for the ISDN line.

I want all my computers to communicate with each other, my company's intranet and the general Internet. I want to use the IP addresses that come with the ISDN line for my internal network.

Subnets and Routing

The first thing I did was partition my network into subnets. Forget anything you've heard about different classes of IP addresses; they're not relevant for home networking. The issue here is subnetting. The idea is that you have a block of contiguous IP addresses, and you want to split them into separate smaller blocks. The only restriction is that the number of addresses on each block must be a power of two.

When subnetting, the first address of each block is called the “network” address. Don't use it for anything. The last address of the block is the “broadcast” address. This, also, is not used for any particular machine. The “netmask” is 255.255.255.*x* where *x* is 256 minus the number of addresses in the subnet.

IP routing is handled as either a network or on an address-by-address basis. For a network, you specify the network address and the netmask instead of just a single IP number. In either case, you have to tell it how to get to that address. If the machines are on a directly connected network, you can just specify a network interface; otherwise, you have to give it the address of another machine that will do forwarding for you.

For my network, I assigned the 16 IP addresses that came with my ISDN modem as follows: 4 for a subnet connecting the ISDN modem and my main Linux system, 4 for the PLIP subnet connecting my laptop and 8 for the Ethernet connecting my desktop systems. The cable modem provides a single IP address for the Ethernet interface it connects to, while the ISDN modem acts like a router with its own IP address.

How It All Works

I completely rewrote my startup scripts for initializing my network. Probably the best way to understand my network is to step through the initialization script for my main Linux system and explain what it does. Like most systems, my

startup scripts are written for /bin/sh (which is actually bash, but I don't use any fancy bash features).

Since the tools for Linux 2.2 are different from those in Linux 2.0, the scripts check which version I'm running, allowing me to use whichever kernel I choose to boot.

```
# Determine kernel version
KVERSION=`/usr/bin/awk '{printf("%3.3s\n,$3)}'\
/proc/version`
```

Now I go through and set up each of the network interfaces. The loopback device is there for making network connections to my own machine. Every Linux system should already have this configured.

```
# Attach the loopback device.
/sbin/ifconfig lo 127.0.0.1
/sbin/route add -net 127.0.0.0 lo
```

I built all the Ethernet drivers as modules. This allows me easy control over which interface is eth0, as opposed to eth1. You will notice I use options to specify the I/O address of my NE2000 card. This is because I have the unusual combination of an ISA and a PCI NE2000 card, and they use separate drivers.

```
# Set up the cable modem,
:echo setting up eth0
modprobe ne io=0x300,1,1,1
```

The cable modem uses DHCP to determine its IP address, and there's a different version of the DHCP client daemon for 2.2.

```
echo -n "starting dhcpcd"
[ $KVERSION = "2.0" ] && {
        /sbin/dhcpcd.old -r eth0
} || {
        /sbin/dhcpcd eth0
}
echo "
```

The second Ethernet card is connected to the ISDN modem. The ISDN modem acts like a router, so it has its own IP address (.209). I add a route for the subnet to use eth1, and I add another route for my employer's network to be gatewayed through the ISDN line.

```
# Set up the ISDN line
echo setting up eth1
modprobe ne2k-pci
ifconfig eth1 inet 172.25.5.210 netmask 255.255.255.252 \ broadcast 172.25.5.211
route add -net 172.25.5.208 netmask 255.255.255.252 eth1
route add -net 168.159.0.0 netmask 255.255.0.0\
gw 172.25.5.209
```

The third Ethernet interface is for my internal network.

```
# Set up the LAN
echo setting up eth2
modprobe tulip options=11
ifconfig eth2 inet 172.25.5.217 netmask 255.255.255.248 broadcast 172.25.5.223
route add -net 172.25.5.216 netmask\
255.255.255.248 eth2
```

For my laptop, setting up PLIP is just like setting up another Ethernet port. Because I have a separate parallel port with a printer attached, I use modules for both PLIP and LP and specify options to **modprobe** so it gets the right port for each one.

```
# Set up PLIP
# see
# http://metalab.unc.edu/mdw/HOWTO/mini/PLIP.html
echo setting up plip0
[ $KVERSION = "2.2" ] && {
        modprobe lp parport=1
        echo 7 > /proc/parport/0/irq
        modprobe plip parport=0
} || {
        modprobe lp io=0x378
        modprobe plip io=0x3bc irq=7
}
/sbin/ifconfig plip0 172.25.5.213 netmask \
255.255.255.252 pointopoint 172.25.5.214 up
/sbin/route  add -net 172.25.5.212 netmask \
255.255.255.252 dev plip0
```

We will use IP Masquerade to allow us to share the single IP address provided by the cable modem, so we want to load the modules to provide support for masquerading protocols needing special help.

```
# Load IP Masquerade modules
# Note that some of these only exist for 2.2
# kernels
modprobe ip_masq_ftp
modprobe ip_masq_irc
modprobe ip_masq_raudio
modprobe ip_masq_cuseeme
[ $KVERSION = "2.2" ] && {
        modprobe ip_masq_vdolive.o
        modprobe ip_masq_quake.o
}
```

Now for the masquerading, firewalling and such. All of this changed in the 2.2 kernel, but the ideas are the same. First, we set up our forwarding rules. These tell Linux what to do when it receives an IP packet from one machine that is destined for another. Anything from the internal machines destined for the Internet should be masqueraded. We also need to allow forwarding of packets between the different interfaces. What we don't want is to allow connections from the general Internet to be forwarded or masqueraded—coming in from the cable modem should be addressed to the IP number assigned for that interface. Hence, my default policy is to forward, and I have special rules to masquerade packets destined to go out the cable modem or come in from the cable modem.

Note that the order of the forwarding rules is important. The kernel must see the rules for masquerading before the rule for denying forwarding on the interface.

```
# Set up IP Masquerading
# This has to be done carefully to only masquerade
# packets that originate from my network. Otherwise,
# someone on the outside could route through my
# system to hide their identity
echo Setting up IP Masquerade
[ $KVERSION = "2.2" ] && {
        echo 1 > /proc/sys/net/ipv4/ip_forward
        ipchains -P forward ACCEPT
        ipchains -A forward -i eth0 -s \
172.25.5.216/255.255.255.248 -j
        MASQ
        ipchains -A forward -i eth0 -s \
172.25.5.212/255.255.255.252 -j
        MASQ
        ipchains -A forward -i eth0 -j DENY
} || {
        /sbin/ipfwadm -F -p accept
        /sbin/ipfwadm -F -a m -S \
172.25.5.216/255.255.255.248 -D 0.0.0.0/0 -W
eth0
        /sbin/ipfwadm -F -a m -S \
172.25.5.212/255.255.255.252 -D 0.0.0.0/0 -W \
eth0
        /sbin/ipfwadm -F -a deny -W eth0
}
```

Now, I like to run X on several machines, connecting to them from accounts on remote Internet hosts. The X server accepts connections on port 6000+*n*, where *n* is the number following the colon in the DISPLAY variable. Hence, I want to forward connections to port 6001 to port 6000 on an internal machine, doing a sort of reverse IP masquerade. With 2.2, I use port forwarding. With 2.0, I use a user-space daemon, since port forwarding isn't part of the standard 2.0 kernel. Note this is insecure, since the internal machine thinks all the forwarded connections are coming from my other Linux box.

Listing 1

This gets a bit complicated, because the normal way of doing port forwarding requires you to specify the IP address the packet was addressed to. Since my IP address is dynamically assigned, I wanted to keep my rules independent of my IP address. Hence, I do a two-step process using “firewall marking”. I create a marking rule that assigns a numbered mark to any incoming packet for port 6001, and then I do port forwarding based on that mark. (See Listing 1.)

Listing 2

The last step is adding some firewall rules (see Listing 2) to block out the general Internet connections I don't want. For example, I run Sendmail, but only for sending mail, as I receive mail elsewhere. Instead of keeping up on all the security issues concerning Sendmail, I just block any attempts to connect to it from outside. You can use **DENY** or **REJECT**. The difference is that **REJECT** will send back a packet saying “connection refused”, while **DENY** will totally ignore the connection attempt.

Now we have a problem with the ISDN modem, because when it sends packets out for my various machines, it thinks they're all on one big subnet. It doesn't realize it needs to send everything to my central Linux system, which will then route things. Since I didn't want to bother the people who configured the modem, I instead used bridging to solve the problem. Bridging is somewhat like routing, but at a different level. Routing uses IP addresses; bridging uses Ethernet addresses. When bridging is turned on, Linux figures out the Ethernet addresses of all the systems on each interface, and if it sees a packet addressed to a machine on a different interface from that machine, it retransmits the packet on the right interface. What is important is that a bridge transparently connects Ethernet segments. It works at the Ethernet level, so any IP-level stuff (such as firewalling, masquerading and subnetting) is completely bypassed. You don't even have to assign IP addresses to the interfaces when bridging. In order for it to work, you have to put the interfaces into “promiscuous” mode, so they will listen for Ethernet packets which aren't addressed to them.

```
# Set up bridging so that incoming ISDN packets
# that don't know that
# I'm acting as a router will be intercepted and
# passed on
ifconfig eth1 promisc
ifconfig eth2 promisc
brcfg -ena
```

Listing 3

Bridging works at the Ethernet level, so PLIP, which is an IP-level interface, can't be a part of a bridge. But, I still want to do the same thing for my laptop. The solution is to use proxy ARP (see Listing 3). The idea is to use the “address resolution protocol”, which maps between IP addresses and Ethernet addresses, to tell the ISDN modem my Linux gateway is supposed to receive packets for the IP number I've assigned to my laptop. Think of ARP as the glue that connects Ethernet to IP networking.

For the client systems, I configure their network interfaces the same way I configured the corresponding interface above, only changing the IP address. I then add a route for the subnet and set the default route to be my Linux box. It also works fine if the client is running MacOS or Windows. Here's an example:

```
# Configure network interface and routes
ifconfig eth0 inet 172.25.5.218 netmask \<\n>
255.255.255.248 broadcast 172.25.5.223
route add -net 172.25.5.216 netmask \<\n>
255.255.255.248 eth0
route add default gw 172.25.5.217
```

Of course, I also had to recompile my kernel, explicitly enabling every feature I used. Some of them are flagged as “experimental”, but they have been completely stable in my experience (although you do need to tell the kernel to allow “experimental” features during configuration). The only networking kernel

patch I've used is an update of the version of the Tulip card driver, because my card is rather strange.

Be careful, though: even people who really know what they're doing can make mistakes. If you find any security holes in my network, please let *Linux Journal* know, so others won't make the same mistake.

Preston F. Crow (Preston.F.Crow@Dartmouth.edu) lives with his wonderful wife in his new house in Ashland, MA. He works for EMC, writing software to run multi-terabyte storage systems.

Archive Index Issue Table of Contents

Advanced search

Assessing the Security of Your Web Applications

Nalneesh Gaur

Issue #72, April 2000

An outline of key test areas to identify security issues in a web application and provide measures to minimize them.

Web sites are moving away from static HTML to dynamic interactive web applications. It is the dynamic, interactive web application that is making the Internet the universal medium. Web applications bring a new level of risk to web sites. Security of these web applications is paramount to the security of the site.

Awareness of security threats from the Internet is increasing the adoption of secure technologies. Deploying firewalls is a standard first step adopted by many organizations. Firewalls protect against many attacks on the network and system infrastructure. In addition, some firewalls provide filtering capability and contain inbound malicious Java and Active-X applications. However, firewalls do little to protect against inbound malicious requests to legitimate applications. Web-based applications are very popular due to the ubiquity of the Internet. Providing access to customer information, user profiles, financial records and health records are common examples of services that web applications can provide. Most often, these applications access a back-end database to serve dynamically generated content to the users. Applications designed without security in mind may result in loss of data integrity, availability, confidentiality and privacy.

Most web application testing can be classified as static or dynamic. Static testing involves manually inspecting the source code and automatically testing for dangerous constructs. On the other hand, dynamic testing involves executing the web application to detect anomalous behavior on unexpected inputs. The focus of this article is on dynamic testing.

The Malicious Intent

Some information seekers think maliciously. Hackers are sometimes able to anticipate inadequacies and the coding practices adopted by programmers. Often, the “speed to market” attitude pushes application developers to overlook standard and secure coding practices. This is especially true in the e-commerce environment, where standard practices such as Change Management are often overlooked. Security is thus usually an afterthought. Often, this results in a vulnerable first release of an application. The process of fixing the vulnerabilities is a fairly expensive one.

Code templates and examples in different development environments provide developers with an approach to implement the desired functionality. These code snippets may not, however, account for application security. The malicious user is sometimes able to identify the development environment just by viewing the HTML code generated by a web application. Comments and some HTML tags can provide information on the development environment. Upon identifying the development environment, a malicious user is able to exploit vulnerabilities where the example or template may have been used.

Another common area for exploitation is the way the application maintains session state information. The HyperText Transport Protocol (HTTP) by itself is stateless. Cookies are commonly used to maintain state information between subsequent HTTP requests. Cookies are simply sets of strings written to the browser by the web application server. They are used to maintain session state, remember passwords and user names, for personalization and configuration features. A malicious user could hijack applications that do not implement strong session controls.

Application vulnerabilities are important because they give access to confidential information such as credit card numbers, account numbers or names and customer lists, without having to break into the web server. The difference between a malicious user and a regular user is *intent*.

Are You At Risk?

In a recent incident that stunned the on-line community, a hacker posted up to 25,000 stolen credit-card numbers on a public web site (see Resources). These numbers were stolen from the CD Universe web site. The hacker claims to be in possession of more than 300,000 credit card numbers from this site. Further, the hacker claimed that the credit card numbers were compromised due to a

flaw in the software used to process credit card transactions. Are you at risk? It depends. You may be at risk if:

- You are a large corporation that attracts many users to your corporate web site.
- You have just released a statement boasting about the security of your site.
- You are completing and releasing a new product in the marketplace.
- You are a financial institution.
- You are a government organization.
- You are a provider of many knowledge-related or data services.
- You are an e-commerce site.

If you do not fit any of the above categories, you may still be vulnerable. In the event of a compromise, only your organization's data classification policy and the value of the data lost will determine the extent of the damage. Just last year, the numbers of web site defacements rose over 900% (see Resources). This can result in embarrassment and unnecessary media exposure. Some publicly traded companies have seen their stock value go down as a result of a breach in the security of their web site.

Securing the Applications

Now we know in theory how a malicious user thinks. We also have some idea about who may be at risk. The remainder of this article will focus on security issues and measures that can be implemented by web site developers in protecting an organization's assets. This is important because the web site administrators cannot easily enforce the client-side security measures. What follows is a list and description of security-related exposures.

Cookie Poisoning

A cookie is a small piece of data which is sent from a web server to a web browser when that browser visits the server's site. The cookie is stored on the user's machine, but it is not an executable program and cannot do anything to that machine. However, cookies may allow a malicious user to hijack web sessions and view, modify or otherwise exploit the information related to another user's session. A hacker may obtain the cookie by various means, including physical access or network sniffing, as well as guessing the cookie's contents. Next, the hacker can try to impersonate the user by hijacking the user's sessions. This is an especially serious issue with shared workstations, cyber cafés and public kiosk environments.

Sometimes cookies are used to store information such as user host name, password, account ID, session ID and other user profile information. Cookies are often used to maintain session information between the user and his shopping cart. Two types of cookies exist:

- **Persistent cookies** have an expiration date and are stored on a user's hard disk until that date. A persistent cookie can be used to track a user's browsing habits by identifying her whenever she returns to a site.
- **Non-persistent cookies** are stored in the web browser's memory. They last only until the browser is closed and are then destroyed.

If a user is able to capture the cookies by sniffing on the network, or by any other means, he may be able to gain unauthorized access to personal information, including credit card number, passwords, user ID and mailing address.

The security measures you can take are:

- Use non-persistent cookies instead of persistent cookies.
- If you must use persistent cookies, then specify a short duration for the cookie's life. The longer the time until cookie expiration, the larger the risk.
- Avoid application features that use persistent cookies to store privacy-related information. Example: “Please check to remember user name and password.”
- Use the secure tag, so that the cookie is sent only if a secure channel (https) is being used.
- Encrypt the information in the cookies. Some web sites split one cookie into many cookies that are further encrypted.

Form Manipulation

Very simply, form manipulation involves saving a web site's form and editing it off-line. Many times, this involves adding more entries to pull-down lists or increasing the size of text fields. The intent is usually to cause a buffer overflow on the server. In the past, client-side form validation has been used to offload the performance load from the server. While client-side validation is a good technique from a performance point of view, it is not the preferred solution from a security point of view. Poorly designed web applications may contain hidden fields that contain user IDs, account IDs or other key fields that define user sessions. Again, all this information can be manipulated off-line to gain access to another user's session.

Form manipulation is a simple technique and requires only a knowledge of HTML. Experienced programmers may be able to alter and submit forms by guessing the server-side code used to process the forms.

The following measures should be implemented to improve the security of an application against form manipulation:

- Perform referrer checks on the server side. This will ensure that a given form was reached from the page that contains the hyperlink providing access to the form.
- Do not rely on the form field-length checks or JavaScript to ensure form input integrity. Perform form input-length checks on the server as well.
- Process and validate the form input field values entered by the user for range, expected input (e.g., numeric vs. alphabets), strange characters and any other associations specific to the user.
- Do not store critical user information in hidden fields in the form.

Bypassing Intermediate Forms in a Multiple-Form Set

Sometimes forms are filled out in sequence. This may be necessary because the information provided in one form is used to take the user to the next form in the web application. Additionally, the entire web application may be divided into multiple forms for ease of use.

Malicious users may be able to bypass the intermediate forms by typing out the entire form name in the browser URL field instead of using the navigation controls provided by the web site pages. This may result in unexpected application behavior, accessing a defunct application, incomplete database records or buffer overflow.

Security measures you can take are:

- Ensure the user progresses to the next form only after all the required information requested in the preceding forms is provided. Furthermore, ensure the user visited all preceding forms.
- Perform referrer checks on the server side. This will ensure a given form was reached from the page that contains the hyperlink providing access to the form.

Embedded Queries to a Relational Database

Many times, form fields are used to send input provided by users to the back end web site for further processing. For example, a user may input their user ID or full name to list information pertaining to their user account. Once the web

server receives this information, it will issue a query to a relational database. The results of the query are displayed to the user.

A malicious user may input field entries in such a way that the returned result provides additional information about other users as well. In addition, the embedded queries may run other SQL commands, such as pipe commands, which may result in disclosure of confidential information.

Security measures you can take are:

- The web application must carefully examine the input fields used to create the database queries for illegal characters, for example an asterisk (*).
- Validate and ensure that input fields contain only the relevant user-related information.
- Ensure proper permissions exist on the database objects accessed by the web application.

Session Timeouts

Many site administrators feel secure simply because the site is using SSL for all its sessions. SSL provides for data transmission security between the web client and the web server. Once an SSL session is established, all information exchanged between the web server and the web client is encrypted. The session timeout specifies the “no activity” duration beyond which the user will have to re-authenticate himself to the web site. The session timeout is usually based on the type of application. Serious financial institutions may specify a very short session timeout period. Regular applications, such as web-based e-mail, may use longer timeout periods.

A malicious user may be able to hijack another user's session if session timeouts are too long. The implications of this are widespread, ranging from embarrassment to loss of confidentiality and integrity of user information. This is a major issue in kiosk, cyber café, laboratory and shared workstation environments.

Your security measure is to evaluate carefully the session timeouts for your application. If you are using multiple application servers, then ensure the session timeouts for the multiple applications are consistent with the timeouts determined for the entire web site.

Directory Listing

Most servers are configured with automatic directory listing. This means any directory that does not contain any of the default files (for example, index.htm

or default.htm) served by the server will display the contents of the directory. This is dangerous for directories where CGI program sources or executables reside. Further, these directories may contain other files (for example, files with the ~ prefix or the .bak suffix) that may provide more information on the web-site application.

Malicious users may be able to browse the directories and download key files. Files that contain source code may be examined to identify trap doors to gain access into the web server or applications.

The security measures you can take are:

- Configure the web server to specify all default files that may be used and to disable directory browsing.
- Establish proper procedures when adding the web-application files.
- Ensure that unnecessary files are periodically removed.

Summary

Dynamic content on web sites will continue to enhance the business functionality of web sites; it is supported by a growing number of e-commerce sites. Also, these web applications are increasingly connected to databases that were previously accessible only through internally built custom applications. Malicious individuals can exploit these web-based applications to gain access to privileged information. Several simple methods, such as cookie poisoning and forms manipulation, can be used to exploit poorly designed web applications; most often, just a text editor and a browser are sufficient. The tools used to execute the exploits are easily available and require minimal knowledge. The very same tools and methods may be used to test the robustness of web applications.

An exhaustive testing of web applications will require building test scenarios to identify vulnerabilities. Proper web-application designs, web-server configuration, secure programming practices and good housekeeping are necessary for the security of any web site and a site's privileged resources. Due to the custom nature of web applications, they pose a challenge to the security of web sites. In the future, web applications are expected to be more secure, as certified components used to build applications gain support. For now, we will have to rely on both static and dynamic testing of web applications.

Resources

Nalneesh Gaur (Nalneesh.Gaur@gte.net) is a manager in the eRisk Solutions practice of Ernst & Young LLP in Dallas, Texas. He has specialized in UNIX and

Windows NT systems, integration and Internet/intranet security issues for a number of years.

Archive Index Issue Table of Contents

Advanced search

Setting Up a Linux Gateway

Lawrence Teo

Issue #72, April 2000

Setting up a Linux gateway can be a rewarding experience in both home and commercial environments.

Networks are extremely common these days—you see them in businesses, schools, even homes. Networking is popular because it allows users to share resources. You can share files, printers and a myriad of other devices in a network. Now, wouldn't it be great if you could share an Internet connection? With Linux, you can.

Setting up Linux as an Internet gateway is not difficult to do. A Linux gateway allows two or more computers to use the Internet at the same time. While doing so, only the gateway's IP address will be visible on the Internet. The rest of the computers will be “hidden” behind the gateway. This is called IP masquerading.

What can you do with this setup? Well, if you have four computers connected to the gateway, you can surf the Web from any of the four computers at the same time. You can run telnet sessions, go on IRC (Internet relay chat), read newsgroups, etc.—*almost* anything you can do on the Internet can be done. Of course, there are certain things that may need your attention, and I will discuss them as well as setting up both Linux and Windows machines to use the gateway.

What You Need

First of all, you need a working TCP/IP network. I assume your network is up and running, and all your machines are able to “see” each other.

I will discuss setting up IP masquerading using Linux kernel 2.2.*x* and ipchains 1.3.*x*. If for some reason you are running an early kernel such as 1.*x.x*, please

refer to Chris Kostick's articles on IP masquerading in issues 27 and 43 of *Linux Journal*.

Also, please make sure you have a copy of the *Linux IP Masquerade mini HOWTO* (http://ipmasq.cjb.net/) by Ambrose Au and David Ranch. It contains much more detailed information on setting up IP masquerading, including setting up Macintosh and Windows NT clients. It also contains a useful FAQ should you run into problems. This article is based on that mini HOWTO as well as personal experience.

I also assume you are familiar with basic Linux system administration, and that you know how to recompile your kernel and modify your **init** scripts.

What Do You Want to Do?

The next thing to figure out is what you want to do. How many machines are on the network? Which machine do you wish to set up as the gateway? Which machines will be the clients? What operating system is each machine running? The answers to these questions can be complex and unique, so for the purposes of this article, we will use the setup shown in Figure 1. This is a three-node network with a Linux gateway (antioch), a Linux client (nazareth) and a Windows 95 client (lystra).

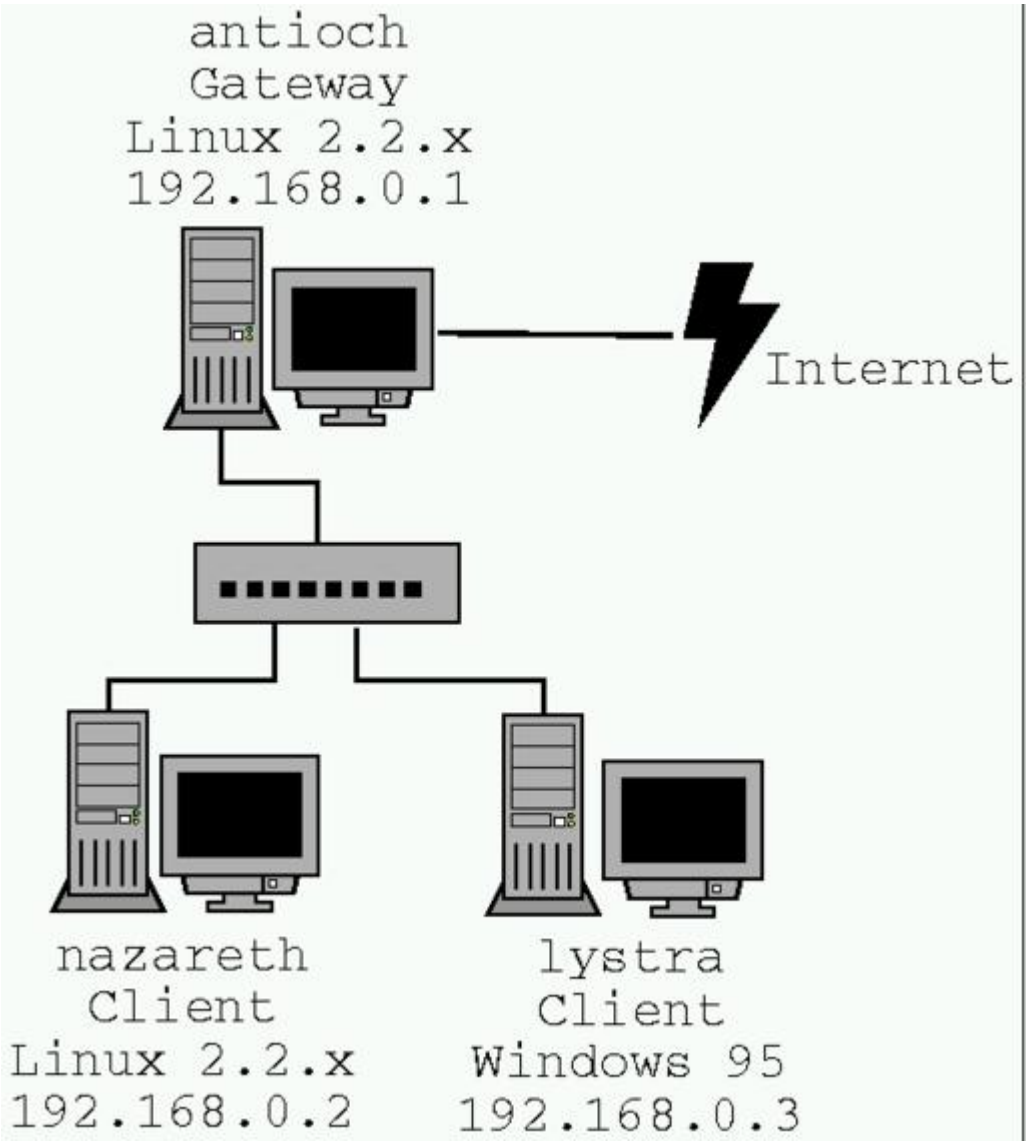


Figure 1. Gateway Setup

Setting Up the Gateway

Let's start by setting up the gateway, which in our case is antioch (192.168.0.1). Antioch runs Linux 2.2.*x*, and in order for it to become a gateway, we need to turn on certain kernel options. My gateway has the kernel options shown in Table 1 turned on.

Table 1

After booting our newly compiled kernel, we will have to load a few kernel modules using either **insmod** or **modprobe**:

```
/sbin/insmod ip_masq_user
/sbin/insmod ip_masq_raudio
/sbin/insmod ip_masq_ftp
/sbin/insmod ip_masq_irc
```

It would be wise to add these lines into one of your init scripts so they will run on every startup. There are other kernel modules related to IP masquerading; for a full list, type the command

```
/sbin/modprobe -l | grep ip_masq
```

Linux 2.2 does not turn on IP forwarding by default. To find out whether IP forwarding is switched on, check the contents of the file /proc/sys/net/ipv4/ip_forward. If it is **0**, IP forwarding is off; if **1**, it is on.

```
# cat /proc/sys/net/ipv4/ip_forward
0
# echo "1" > /proc/sys/net/ipv4/ip_forward
# cat /proc/sys/net/ipv4/ip_forward
1
```

Again, it is wise to add the line which turns on IP forwarding (the one with the **echo** command) to one of your init scripts.

Now we come to an interesting situation. How do we know who gets to use the gateway and who doesn't? This is where **ipchains** comes in. My current policy is to deny access to the gateway from everybody unless explicitly allowed. For example, let's say we want only our client machines nazareth and lystra to access our gateway and no one else. In order to do this, we have to issue the following commands:

```
ipchains -P forward DENY
ipchains -A forward -s 192.168.0.2/255.255.255.0\
  -j MASQ
ipchains -A forward -s 192.168.0.3/255.255.255.0\
  -j MASQ
```

If, on the other hand, we want everybody on the network 192.168.0.* to use the gateway, we can issue these commands:

```
ipchains -P forward DENY
ipchains -A forward -s 192.168.0.0/255.255.255.0\
  -j MASQ
```

Note that we assume the netmask is 255.255.255.0. If your netmask is different, simply change the values accordingly. There are many other things you can do with ipchains; however, they are beyond the scope of this article. I trust that the two simple examples above will get you started. (See also “Building a Firewall with IP Chains” by Pedro Bueno, http://www.linuxjournal.com/lj-issue/issue68/3622.html.)

That's it! The gateway is now up and running. Remember to add the relevant lines to the startup scripts. Also remember to connect to the Internet before testing your gateway. Now let's set up the clients.

Setting Up the Linux Client

Setting up the Linux client (nazareth, 192.168.0.2) is very easy. All you need do is issue the following command on nazareth:

```
route add default gw antioch
```

Now try pinging an external site (let's say www.ssc.com) to see if it responds:

```
ping www.ssc.com
```

If it responds, you are in business! If it doesn't, check the FAQ included with the mini-HOWTO for solutions to frequently encountered problems.

Setting Up the Windows Client

Setting up the Windows client is a bit more troublesome. Here are the steps involved:

1. Go to the Control Panel and double-click Network.
2. Locate the icon that represents your TCP/IP protocol for your network interface card. Open up its Properties.
3. Click on the Gateway tab. Add 192.168.0.1 as the gateway.
4. Click on the DNS Configuration tab. Under DNS Server search order, add your ISP's DNS server IP addresses.
5. Press OK on all the dialog boxes.
6. Reboot the machine.

Again, test your gateway by accessing an external site (use **ping** or your web browser or whatever). If all goes well, you should be able to do most things you usually do on the Internet.

Precautions

There are a few things you should be aware of when setting up your Linux gateway.

First of all, certain Internet applications may not work well with our setup. For a list of what works and what does not, see the latest version of the IP Masquerade mini HOWTO.

A few applications may require you to load specific kernel modules. In our case, for example, we have already loaded **ip_masq_raudio**, which will take care of any Real Audio connections. If you want to run Quake, VDOLive or CUSeeMe, you will need to load their respective kernel modules.

Another thing to keep in mind is that applications on your Linux client machine may not work properly if your gateway is not connected to the Internet. One such application may be sendmail. Therefore, if you know your gateway is off-line, you may want to remove your gateway's IP address from your Linux client's routing table. To do so, just issue the following command on the Linux client machine:

```
route del default
```

Conclusion

A Linux gateway offers a great solution to using and sharing a connection to an external network. Linux is extremely suitable for use as a gateway for both home and commercial networks because it is low in cost and reliable.

email: lawrenceteo@usa.net

Lawrence Teo (lawrenceteo@usa.net) recently completed his Bachelor of Computing degree from Monash University, Australia. He has been using Linux since 1997 and has been glued to it since. His other interests include security, cryptography, webmastering and software development. Lawrence aspires to be a UNIX system administrator one day.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Linux and the Next Generation Internet

Michael Stricklen

Bob Cummings

Stan McClellan

Issue #72, April 2000

The authors describe their implementation of a demonstration environment for differentiated Internet services (Diffserv) using Linux-based routers.

Our Diffserv implementation is relatively simple, but highlights many of the strengths and flexibilities of Linux, which we believe are especially important in the era of Internet-centric computing. To make this a more complete description of the several technologies related to “differentiated services”, we include some background material on the purpose and structure of Diffserv as well as highlights of other, more complex implementations of Diffserv environments which have benefitted from the sophistication of the Linux kernel.

Although our implementation of a Diffserv environment is fairly straightforward, we feel it's also interesting for these reasons:

- We use an approach that allows reconfiguration of the entire network instantaneously and on-demand from a “network management workstation”. This was a key component of our environment, because one of our main goals was to demonstrate concisely, for a non-technical audience, using real-time applications, the “before-after” effects of a Diffserv-enabled network.
- We developed this environment supported, in part, by one of the major Regional Bell Operating Companies (RBOCs) and by the National Science Foundation through a “Research Experiences for Undergraduates” (REU) supplement to an existing NSF grant. As such, the system configurations, software and most of the architecture was developed primarily by undergraduate engineering students at our university.

Our demonstration environment has been used for live, hands-on demonstrations at two large regional meetings: the Southeastern Universities Research Association (SURA) Applications Workshop (Sept. 1999) and the Bellsouth Science & Technology *Innovations Showcase* (Oct. 1999).

Diffserv: Some Important Background

The key to "service differentiation" (or Quality of Service, QoS) in the Internet is the way routers handle (or can be easily modified to handle) multiple classes of traffic with various requirements for transport. The term "Diffserv" refers to an approach for implementing such capabilities which is being defined (through the usual method of constructing Internet standards) to be broadly compatible with the scope and flavor of the global Internet (see Resources 1).

The architecture of Diffserv can be viewed in terms of relatively simple functional units in the Internet's "forwarding nodes" (routers) (see Resources 2). The simplicity of Diffserv is important because, in theory, it has the potential to provide coarse differentiation between types of Internet traffic without requiring a fundamental change to the current configuration of the Internet.

One of the functional units described by Diffserv is a set of "per-hop behaviors" (PHBs). The idea behind PHBs is to let each router easily and quickly classify packets into different types of output queues based on a "tag" embedded in the packet header. Square tags go in "square" queues. Round tags go in "round" queues. Packets in the "square" queue get treated differently than packets in the "round" queue.

The scheme works in much the same way airline passengers are allowed to check bags or board the plane: "first class" goes first, "coach class" is next, and "standby" is last if there's enough room. There are also other functional units in Diffserv which are often called *packet classification* and *traffic conditioning*. In keeping with the airline analogy, packet classification is akin to the act of purchasing a type of ticket (or having one assigned to you, based on some rules) and traffic conditioning is like the disturbances (e.g., bumping and rerouting) experienced by a group of passengers when a flight is canceled or delayed. For our demonstration environment, we focus primarily on the PHBs and differences between particular classifications of traffic when the network is congested. To put it another way, we essentially ask the following questions: "Is first class *really* better than coach?" and "How can I tell?"

In Diffserv, the "first class" designation is called *expedited forwarding* (EF) (see Resources 3). The idea of EF is to simulate a "virtual leased line" by ensuring minimal queuing of packets within each router along the transport path. As such, the EF class hopes to provide *guarantees* on delay and jitter, which are

important for isochronous data streams (i.e., video and audio). This is one of Diffserv's weak points, in our opinion. Due to an explicitly designed inability to distinguish between individual traffic streams, the *aggregate* EF flow receives the desired treatment. There can be no “hard promises” made to individual flows unless there are *very few* EF flows. This result has been noted with some chagrin in many publications (see Resources 4). The effect of EF classification is the presence of high amounts of jitter between subsequent packets in individual streams. As a result of the stated goals and the architecture of Diffserv, the only way to minimize these effects is to practice “gross overprovisioning”, where only a small percentage of the available bandwidth is made available to the EF class, and only a few EF streams are allowed. In the airline analogy, the number of first-class passengers on any flight would have to be limited strictly to a tiny proportion of the available seats. Otherwise, the flight attendants wouldn't be able to *guarantee* good service.

The “coach class” designation in Diffserv is called *assured forwarding* (AF) (see Resources 5) and is a bit more complicated than EF. The complication of AF is primarily due to the fact that there are four different classes of AF, and each class has three subtypes. The difference between AF classes is related to different levels of “forwarding assurances”. The difference between subtypes in each AF class is related to different levels of “drop precedence” or relative importance within the class (i.e., low, medium, high).

The relationship between “class” and “drop precedence” is subtle. Each class is allocated resources (such as buffer space, bandwidth and so on) at each forwarding node (router). These resources comprise a level of “assurance” that packets from each class will be forwarded as desired. Transmissions can *exceed* these resources at their own peril, described by the “drop precedence”. So, within the AF designation, forwarding depends on the relationships between the instantaneous traffic load at a router, the “available” resources compared to the “desired” resources and the drop precedence of each packet.

The “standby class” designation in Diffserv is the well-known *best effort* (BE) behavior of the current Internet. So, coarse differentiation between service levels is made by classifying packets as BE (poor), AF (better with conditions) or EF (best).

As a result of the functional unit architecture of Diffserv, and in an effort to push per-stream complexity to the *edge* of the network, there are actually at least two different types of routing/forwarding nodes in a Diffserv domain. According to the Diffserv specification, “edge” routers use a (possibly complex) set of rules to insert tags into the header of each IP packet. These tags are called “Diffserv Code Points” or DSCP (see Resources 6). Once the packets have been tagged and admitted into the interior of the Diffserv domain, “core”

routers simply have to examine each packet's DSCP and assign it to the corresponding output queue to be forwarded on to the next node. With proper network architecture, each packet should be able to consume the forwarding resources it needs and is entitled to as a result of its “tag”.

Linux Support for Diffserv Capabilities

The ability to implement advanced routing behavior using Linux, including those proposed by Diffserv, is provided by the rich set of traffic-control features present in the Linux kernel. Alexey Kuznetsov is the author of these kernel features and the user-space programs used to control them. The architecture of the Linux traffic control features is described nicely by Almesberger (see Resources 7), and the motivation and control of these features is also summarized in an excellent *LJ* article by Hadi-Salim (see Resources 8). For clarity, we include a brief review of the Linux traffic-control capabilities used in our implementation and our approach to configuring them. In general, to enable “differentiated services” for Linux, first the Linux box has to be able to route IP packets correctly, and several rules for traffic control must then be put in place.

Kernel Configuration

In preparation for use as a Diffserv router, the kernel of the Linux router must be configured to allow the use of advanced routing features. To implement Diffserv-type behaviors effectively, several “subsystems” of the kernel must be available. These subsystems include the routing capabilities of the kernel, the packet scheduling functions, and the netlink functionality to configure the traffic-control modules. The traffic-control functions can be compiled into a monolithic kernel or loaded as modules.

Listing 1

Listing 2

A summary of the pertinent features compiled into our Diffserv routers is shown in Listings 1 and 2. All locations given are representative of the option list given during **make menuconfig**. You may be checking your kernel configuration menu now, and saying to yourself, “Hmm... I don't see those choices!” That's because you haven't acquired the necessary kernel patch. The web site for “Differentiated Services on Linux” is maintained by Werner Almesberger at the Swiss Federal Institute of Technology (see Resources 9). Here you will find the “Diffserv for Linux” distribution (as of this writing, the current version was ds-6). The distribution comes with a set of patches for both the kernel and for a user-space application to configure traffic-control kernel features (called “tc”). Also included in the distribution is a set of example scripts

and some documentation. It is a good idea to acquire a copy of the package **iproute2+tc** at this point (see Resources 10). The patch from the Linux Diffserv distribution is version-sensitive with iproute2+tc, and since our project took place mainly in the summer of 1999, we used version ss990630 of iproute2+tc.

Once your Linux router has been configured properly (depending on your router's job), you are ready to configure your machine for traffic control.

Traffic-Control Configuration

To enable differentiated services on a Linux router, the traffic-control features must be configured. This configuration is achieved through a user-level program, appropriately named **tc** (traffic control). The command-line syntax for tc is quite long and complex, so scripts are generally used for configuration. An example tc configuration script is shown in Listing 3. In the listing, tc is being used to configure kernel traffic control for a core router in our Diffserv application. This entails attaching a parent queuing discipline to the applicable interface, then creating the queues for the varying classes of traffic. Finally, filters are created to classify packets into the appropriate classes.

Listing 3

As can be seen in Listing 3, the structure of the tc configuration scripts for a Diffserv-enabled Linux router can be broken down into parts:

- Creation of the root queuing discipline. This uses the syntax **tc qdisc add** followed by several parameters. These parameters describe attributes of this queuing discipline. These parameters include which network interface the queuing discipline is attached to (**dev eth3**), an identifier for **qdisc** (**handle 1:0**), where in the qdisc hierachy to insert this qdisc (**root**) and which queuing discipline to use (**tcindex**). The remaining parameters are specific to the particular queuing discipline. Diffserv maps naturally into a class-based queuing scheme. Therefore, each Diffserv router (regardless of job) will employ class-based queuing (**CONFIG_NET_SCH_CBQ**) to house its various per-hop behaviors.
- Creation of classes for each type of per-hop behavior. This uses the syntax **tc class add** followed by several parameters. These parameters are similar to the **tc qdisc add** syntax. These parameters will identify which queuing discipline the class belongs to, and other parameters define the behavior of the class. Our demonstration made extensive use of two per-hop behaviors: best effort (BE) and expedited forwarding (EF). The configuration in Listing 3 clearly shows the two sections defining BE and EF PHBs.

- Creation of queuing disciplines for each class. Each class must have a queuing discipline to determine how packets are enqueued and dequeued. The syntax for this step is identical to that for step 1. The EF PHB class uses a simple FIFO (first-in, first-out) for its queuing discipline, since we wanted the traffic to get in and out of the class as quickly as possible. The BE PHB class uses a token bucket filter in an attempt to throttle the traffic-generation machines during times of extreme congestion.
- Creation of filters (classifiers) to assign marked traffic to the appropriate class. This uses the syntax **tc filter add** followed by several parameters used to describe which packets are bound for what classes. Our sample script is from a core router. Packets arriving at this interface have already been marked by edge routers. Classifying packets at this step requires matching the TOS (type of service) bits from the IP header to values suggested by the IETF (Internet Engineering Task Force) Differentiated Services workgroup for various per-hop behaviors (denoted by the value following the "mask" parameter). The filter creation varies, based on which job the router fulfills. Core routers solely use the **tcindex** packet classifier (**CONFIG_NET_CLS_TCINDEX**) included with the Diffserv distributions. Edge routers use the firewall packet classifier (**CONFIG_NET_CLS_FW**) along with **ipchains**.

Complete Diffserv functionality really assumes two different types of routing capabilities: "core" and "edge" routers. With a Linux-based Diffserv implementation, "edge" routers use ipchains to handle their tasks. Replacing the application **ipfwadm** from earlier kernels, ipchains is a user-space program that configures the firewalling functionalities of Linux kernels 2.1.*x* and higher. Configuring ipchains has been well-documented in this magazine (see Resources 11) and other arenas, and is beyond the scope of this document. Our Linux Diffserv testbed uses ipchains to assign handles to incoming traffic based on IP address rules. These handles are then used by a filter (classifier) installed with tc (the user-space application) to replace the current IP TOS bytefield setting with the appropriate Diffserv field marking (DSCP). This method proved to be very effective. Dynamic configuration was easily attainable, and the speed of ipchains held up to very high demand. Even though ipchains will be superceded by **iptables** in future versions of the Linux kernel (see Resources 12), the functionality will be very similar. So, the approach we've used will still be applicable.

The specific scripts used to provide Diffserv capability in our testbed environment are available at ftp://cter.eng.uab.edu/Diffserv/.

A Diffserv Environment

The ability to change router configurations on the fly in our “demonstration environment” is achieved via a web-based network management console which uses JavaScript bundled with Dynamic HTML. The network administrator can interact with this interface to initiate various levels of traffic priority. We use this to simulate a service-level agreement (SLA) between the network provider and the end user.

A goal of our demonstration environment, in addition to concisely demonstrating the effect of differentiated services, was to prove that the queuing mechanisms within the Linux Diffserv implementation were robust enough to enforce various SLAs throughout our Diffserv domain. As shown in Figure 1, the domain was composed of three routers (one core router, two leaf routers), two Litton CAMVision-2 MPEG-2 codecs (up to 15Mbps) or two Vbrick MPEG-1 codecs (up to 3Mbps), two client workstations, one web server and one network management workstation (NMS).

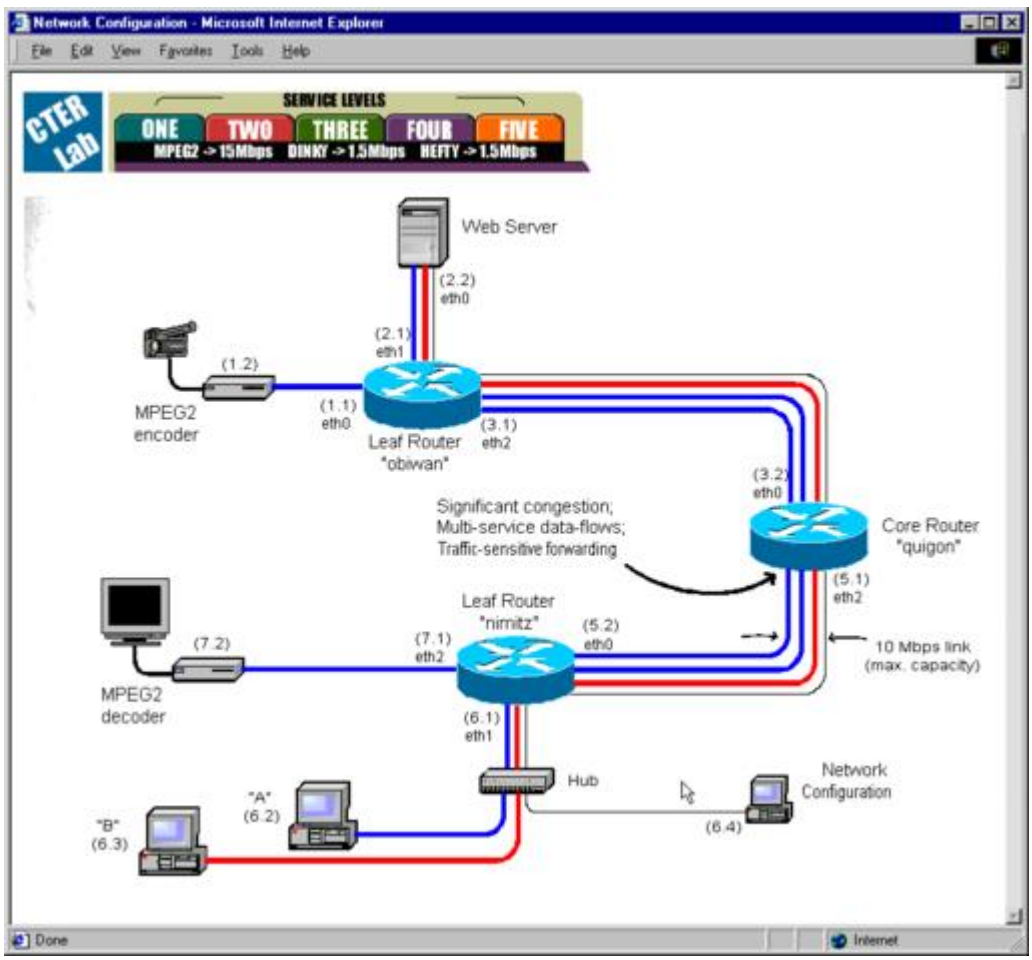


Figure 1. The General Architecture of Our Diffserv Domain and a

In the figure, the classification of traffic is performed by the leaf routers “obiwan” and “nimitz”, and the core router “quigon” is configured for the corresponding DSCP-based forwarding and queueing. The traffic streams are

color-coded to correspond to particular types of PHBs (blue=BE, red=EF and so on). Notice from the figure that the link between quigon and nimitz is 10 MBps Ethernet and is consistently *oversubscribed* with multiservice traffic. This is the situation where differentiation between SLAs is critical. To make sure the instantaneous change between SLAs was clearly visible to the casual observer, we used the MPEG video stream as well as some interactive, web-based streaming media (RealAudio, RealVideo, etc.).

In order to simulate the dynamic nature of the signaling between the network administrator and each of the routers, we decided to use a socket interaction. When the network administrator wishes to configure a certain service level, he accesses the web interface via the NMS. By using JavaScript, DHTML and GIFs with transparent sections within the page, we were able to present the administrator with a visual representation of the desired SLA before actually committing to it. This is shown in the screenshot of Figure 1 as the collection of color-coded traffic streams between the end points.

As shown in Table 1 and Figure 1, we were able to configure several service levels with our approach, each of which was available via a single mouse click. Note that the values and configurations shown in Table 1 and Figure 1 reflect a particular set of SLAs which used only BE and EF traffic classes. When the user clicks on the desired SLA icon, the value from the HTML form field is passed to the web server via an HTTP **POST** operation. The form values are passed via CGI to a Perl script that processes the POST, then reconfigures each router in the domain. The routers are contacted one by one, and the SLA chosen by the administrator is invoked. Sample Perl pseudocode for the client portion of router control is shown in Listing 4, and the server portion is shown in Listing 5. As can be seen from the Perl client code in Listing 4, the NMS (or other web server) can easily pass the “current SLA” to all routers in the domain based on input from the network manager. This “control channel” interface was protected in all network configurations by a high-priority, low-rate queuing configuration, shown as the black line in Figure 1.

Listing 4

Listing 5

To provide positive user feedback at the NMS, the web interface is refreshed for the administrator while each router begins its unique network setup. Each Diffserv-enabled router in the domain receives the desired SLA and must set up its rules accordingly, depending on its position within the domain and the collection of statically defined SLAs. This is done dynamically via a **system** call to **ipchains-restore** according to the new SLA. When the ipchains-restore command finishes, the network setup is complete. The Perl pseudocode for this

operation is shown in Listing 5 for a typical core router. As our system is defined, we maintain essentially a simple “database” of network/SLA configurations in pre-stored ipchains mappings.

To attempt to simulate some typical end-user traffic in addition to the constant MPEG stream, we used a number of FTP downloads, some streaming audio/ video sources and a small flood ping throughout the network. Due to the interactive nature of our demonstration environment, these network-based data sources were also available “on demand” from a web-based GUI.

Other Linux-based Diffserv Work

It would be inappropriate, particularly with respect to any open-source developments, to neglect mentioning related efforts, or efforts which have contributed to the system described in this article. Two Linux-based Diffserv projects we feel are especially interesting and mature are the efforts underway at the University of Karlsruhe (see Resources 13) and the University of Kansas (see Resources 14). Many of the conclusions and insights made available through these projects correspond with our own observations, and they are excellent sources of further information on Diffserv and differentiated services under Linux. We highly recommend them to the interested reader.

In particular, we want to call attention to the differences between the demonstration environment described in this article and the DiffSpec tool under construction at the University of Kansas. The Diffserv approach to resource allocation for each class of service very explicitly requires external intervention in the form of what has been called a “bandwidth broker” (BB). The DiffSpec tool entails a much grander system concept than the demonstration environment discussed here. For example, DiffSpec includes an API for managing queue/class/filter combinations, CORBA-based system calls for automated configuration of DS parameters, and a general web-based user interface to the Linux traffic-control capabilities.

In contrast, for the purpose of our demonstration environment, we unwittingly followed a “separation of powers” philosophy well known to students of political science. We carefully segregated the “service level definition” (or “legislative branch”) functions of the BB into a manually crafted, static database of allowable configurations. At the same time, we placed the “network instantiation” (or “executive branch”) functions of the BB onto a cleverly distributed arrangement of ipchains rules. In this fashion, we are able to reconfigure the entire network instantly to one of several predefined “looks” through either an operator's input or by an automated means. This approach may be scalable in some contexts, and it may provide for convenient “governance” of network resources, but it was not specifically intended for mass consumption.

Additionally, the structure of the Karlsruhe Diffserv implementation seems to be somewhat different than the implementation maintained by Werner Almesberger at the Swiss Federal Institute of Technology in Lausanne. For our project, we used Almesberger's distribution, so we don't have specific experience with the Karlsruhe distribution or the differences between the two implementations.

Conclusions

In reviewing the architecture and explicit results provided by the K.I.D.S. project (see Resources 4), we agree with their conclusions regarding strengths and weaknesses of Diffserv. In particular, through the use of our “before-after” scenario for configuring a Diffserv domain, we have experienced first-hand corroboration of these factors in the context of our “real applications”.

We also agree with Metz, who states (see Resources 1) that “In the long run it will most likely be a combination (of technologies) that will enable the Internet to offer QoS.” When the long run materializes, we're confident that Linux will be a part of the solution, because QoS in the Internet is definitely “where you want to go tomorrow”.

Acknowledgements

Resources

Michael Stricklen (goose@uab.edu) is a research assistant at the UAB Center for Telecommunications Education and Research. He enjoys tinkering with Linux projects and networking gear. When not in front of a computer, Michael would prefer to be riding a snowboard.

Bob Cummings (bahb@uab.edu) is a research assistant at the UAB Center for Telecommunications Education and Research. In his spare time, he likes to spend time with friends, play RPGs and hone his Perl skills.

Stan McClellan (smcclell@uab.edu) is a Linux enthusiast who happens to be employed in the UAB Dept. of Electrical and Computer Engineering. When he isn't “playing Professor” by hustling research money or teaching classes, he can be found wandering around, pestering students for “interesting results”

Archive Index Issue Table of Contents

Advanced search

Eid Eid, OE/ONE Corporation

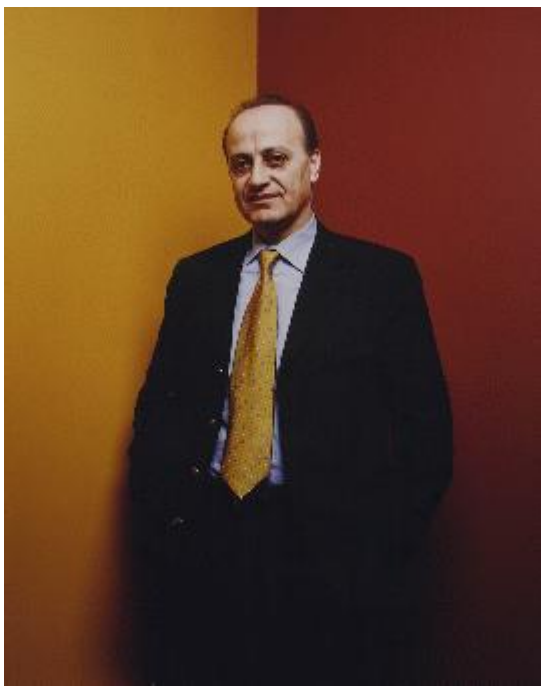
Marjorie Richardson

Issue #72, April 2000

Internet appliances are the next wave of computers to be bought by the public. Mr. Eid tells us how his company is serving this market.

I talked to Mr. Eid Eid back in December of 1997 when he was President of Corel Computer Corporation. Today, he is the head of his own start-up company, OE/ONE (soon to have a name change), dealing with the information appliance market. I talked to him on February 2 to learn how the new company is doing and his take on the future.

Margie: Last time we talked, you were president of Corel Computer. When and why did you leave?



Eid: Well, there were many events that influenced this. When Corel began the computer division, we steered toward a Linux strategy. We had evaluated many real-time and regular operating systems, and selected Linux early in the

process as the right OS for our computers. Eventually, as time passed, we saw Java was not going to deliver on its promises; basically, Sun was very slow at delivering the goods. So, we decided to shift our strategy to building thin-client and thin servers based on Linux, and the NetWinder project was born and was doing very well. Corel at that period had acquired WordPerfect. There were many financial constraints, and Corel decided to spin off their computer division. I left the company then in the early summer of 1998. A few months later, that divesting happened and the NetWinder was sold to Hardware Computing Canada, which later became Rebel.com.

Margie: Right; so then you started the new company?

Eid: No, after that I took a break and did some consulting jobs in Montréal. Then, I joined a start-up here in the U.S. early in '99. It turned out not to be the right environment for me, so I left in June '99 and started OE/ONE.

Margie: When Phil heard you talk at UniForum in '98, you were talking about a fictional character called Mr. Twenty O'One and what his life would be like. I was wondering if your company name, the OE1, came from this character?

Eid: Actually, it may! I mean “one” sounded right, you know. It is only annoying now that so many companies use it. We are thinking of a name change, and in fact, we have a name in mind. If you are interested in the UniForum talk, I can send it to you, it is in the UniForum 98 report. It was in *Linux*--it was in the *Linux Journal*, August 1998! Yes, it was a vision. It was basically an internal paper written for the company that I put on our intranet. I modified it and twisted it into something that would suit the UNIX crowd and also the computing crowd. My vision is actually quite simple. Mr. Twenty O'One is an information worker, a corporate person or a regular consumer who wants a flawless flow of computing access to content and communication. So he holds this hand-held device, I actually had a picture of the device, okay? It looked a little bit like a Pilot—the Palm VII didn't exist then—with a CDMA antenna and a little camera in it. Basically, as he leaves home, a whole chain of events happens which turn on how he deals with content and communication until he gets to work, where he connects this device to his desktop or laptop computer and downloads the information.

Margie: Are you trying to make this scenario a reality for all of us with your new company?

Eid: Yes, in fact, what I said at the end of that speech was that all these technologies are available today—only the integration is missing. The network computing paradigm or Internet, it is the way to go as long as it is served with an infrastructure like Linux, due to its robustness and openness. This is why we merged the open-source Linux with network computing in general.

Margie: Tell us about the products you are building right now: the Web Appliance One and the MMedia.

Eid: In effect, the company is not building those products, we are only building working models of those products which we will offer our customers. So the product as described is in fact a software package, but a software package that you don't see; that is, you will not see it shrink-wrapped on the shelf. I call it OE1, which stands for operating environment, 1 being the number one. Actually, the 1 means two things. The unification, which goes back to my speech at UniForum; and two, it is the first, so there will be a second and a third.

Margie: Okay, so someone else is going to be building the machines, and you sell them the software to put on them?

Eid: That's right. We have two target customers, but in fact they can be two in one. First are the consumer electronics manufacturers, such as Sony, Panasonic, Philips, RCA and the like. All this crowd, the multi-billion-dollar companies, missed out totally on the PC platform, and they have not yet benefited from the Internet era. Here is their opportunity. We believe the market is ready for a machine that looks much more like an appliance than a PC. These manufacturers could sell it like a TV, but it has an advantage. Since it is a smart machine, an Internet machine, they can secure revenues by establishing a link between their customers who become members and themselves. We believe those vendors deserve to own the customer

relationship and the branding, rather than somebody in the middle, like Microsoft, Sun or Oracle. What we charge them for is the software license, the software, professional services to customize it, upgrade and maintenance costs.

Now the second category of customers are what I call FSPs, the full-service providers. Those are the guys like AOL or Earthlink, or they could also be those new emerging high-speed providers, or network service providers, using such things as ADSL or the cable modem. Let's talk about those in particular. Today, when they compete in the marketplace, the only thing they compete with is raw data, raw speed, and they aren't doing a very good job at all. In fact, I myself have a cable modem at home, and the provider wouldn't be able to convince me on speed, because the performance is too variable and not always as smooth as a 56K modem. So what I believe is those vendors do have a value add, but the real value add is in providing a full complete service, including a portal and serving applications and content. What better way of doing it than by delivering a device that is a total solution, rather than having to rely on an installation network to install? That service cost me $150 and I had to do much of it over because the proxy didn't work, it didn't work with Linux, I had to go to a static IP, etc.—you know all these problems. With our approach, the OEMs can merge the cable modem with the machine—we do it for them—and they deliver a full machine for, say, $500. So the customer pays that amount, or in fact, the vendor could subsidize it in the service. For example, instead of charging people $29, they charge them $39 a month, and it covers the cost of the machine.

How do they make money? They up-sell. And the telcos, they do this already. My telco charges me $5.00 a month just for voice mail or Caller ID. Those are very simple applications, but I am still willing to pay $5.00 a month for the convenience. When the OEMs have this connection to their customers, they can then up-sell them on many other applications. It could be a personalized finance system; it could be just for possessing on-line; it could be for extra content, because they have the bandwidth, multimedia content. We want our platform to be thin enough that the OEM can build it inexpensively, but high-end enough that it gives the same experience as an iMac or a Windows 98-based Internet machine.

Margie: Would they be able to add the other software that normally comes on a PC, or wouldn't they want to do that?

Eid: At the base level, you get a first-grade browser, which is Netscape Communicator including Java support, and the basic half-dozen non-standard contents, such as a full PDF reader, a real networks client, real video/audio, an

MP3 player and even a Flash plug-in. I'm not sure yet if you'll download it, or we'll integrate it. So that defines at this level of delivering high-end contents.

Margie: When do you expect to have the software ready and agreements made so that people can actually go out and buy one of these machines?

Eid: Where we are at now is we have a very strong prototype that I'm showing on a notebook and machines at work. We are building half a dozen models. We'll have the first two in a week or so. Those are working models with a neat industrial design, and they are a true Linux platform running the whole software. Most of the user interface for the laptop is pretty much done. We are working on the Linux infrastructure to make it robust and small and on the back end services, such as calendaring, voice mail and the like.

We are targeting a first software cut about four months from now. We worked first with OEMs, but have now completed the sales and business team so that we are starting to work with ISPs to bring everyone to the table.

Margie: Sounds like you are making a good start. What about the MMedia?

Eid: The second appliance, the MMedia, is the high-end one. It assumes the OEM will integrate (which is what we are doing, actually) TV tuners and other media inside the hardware—all integrated, perhaps, on the motherboard. Our software will add the capabilities of what we call personalized entertainment, such as TiVo and Replay. Thus, you can basically flip through channels, record, play back, etc.—we want the machine to look more like a TV, but a high-quality TV, a personalized TV. That is the difference between the two platforms—basically base-line and high-end multimedia.

Margie: There is a company here in Seattle that is putting out an Internet appliance like the one you are talking about. It looks like a thick book. On it they've actually installed a stripped-down version of Microsoft Windows NT, and I was wondering if y'all even considered doing that?

Eid: Well, I've always said I would love to use Windows if it had three elements in it. Number one, if it was modular enough so that you could trim it down, which I don't believe is feasible. Number two, if Microsoft would give us and our customers full access to the source code, and number three, if it was free!

Margie: Oh, right! I don't think that will happen soon. This other company said they would be selling the appliance for around $600. You are beating them on price by about $100. Is that the cost of the operating system?

Eid: Well, Microsoft could decide to bring the price down to $50-$60 if they wanted. In my business stand, this is one of the risk assessments: what if

tomorrow Microsoft decided to make a 90% cost cut on the price of their OS? Then, it would be a threat, yes.

Margie: IDC has been reporting that the information appliance market is going to grow phenomenally in the next few years. I assume you believe this is true. Why do you believe that?

Eid: Well, what I believe is—let's not talk dollars, let's talk numbers—it is only a natural trend. Today, 63% of people still don't have access to computing or the Internet, so the market penetration has been capped below 40% at about 37%. This is true for many reasons: cost, maintenance, deployment, etc. People don't want to mess with the complexities of a fully featured computer. So the 63% of people without access now will be the target market for these appliances. And these people will buy, since the Internet has become as much an entertainment source as one for information. The numbers are predicted to be at least two or three times bigger than the PC market itself. That is not to say the PC market will shrink. By IDC's report, it is still growing at about 9% yearly. I believe that will be sustained for a while, because the momentum behind Windows is tremendous. So, number-wise, we believe with IDC that many more people will be accessing the Internet through these alternative devices than through Windows or Macintosh machines.

Margie: Do you think Linux is going to dominate that market on these devices?

Eid: Linux has the best shot at it. The reason is ubiquity. Many operating systems have claimed the role to unseat Windows, and obviously, none of them could: not the Mac, not OS2, BeOS or NextStep. Only one OS has a chance of doing so—Linux, thanks to its open source. And also thanks to the vote of confidence from the user base and Wall Street. That vote has put enough momentum behind that operating system so that today, you could spend hundreds of millions of dollars to improve it and build applications that run on it.

Margie: It has been fairly amazing.

Eid: Yeah! Don't you wish you had been able to be a part of these IPOs? Many of our friends and former employees work at VA Linux, and it has been good for them.

Margie: I think it is good to see these people, who have been so faithful to the Linux community for so long, actually get something back in the form of financial reward. People can't work for free forever.

Corel also obviously has faith in the viability of this market, since they have acquired a 30% share of your company. Are the machines going to come with Corel Linux pre-installed?

Eid: Absolutely. In fact, we are still finalizing this. I've always said the deal with Corel was not only financial—it was multifaceted; this is because it is a great fit. Mike and I have never stopped seeing each other; we're friends. However, we felt it best to wait until it made sense for Corel and us to enter into a relationship before making a deal. Basically, we are going to base our OS on Corel Linux and the Debian distribution, because they made so many improvements to the user interface, the browsing and other stuff. All of these improvements are very, very useful. So the high-end aspects of our device would have a Corel Linux in it and branded Corel Linux. We don't know exactly how we are going to package it, but it is a certain thing we are going to have Corel Linux in there. Also what we have is access to source code of WordPerfect for Linux, but I can't say much on what we are going to do with this. It is going to be very exciting.

Margie: Sure. Which browser are you planning to put in?

Eid: Mozilla, eventually. For now, we are using Netscape 4.7 until Mozilla releases and we can get on the bandwagon. We will be using all the Mozilla tools and many web-top applications, such as calendaring, a desktop, a file viewer and the like.

Margie: Do you think the predictions you were making back in 1998 at UniForum will become reality before 2001?

Eid: I picked 2001 for many reasons, but the basic one is the wireless. And the wireless has disappointed me in previous years, but now I'm very encouraged after the last couple of shows I've attended. Basically, things are coming together nicely to get the power consumption and the price we were waiting for, so it's about to happen. What is not happening yet is a basic digital PCS Personal Computing Services) that fits in DMA that is low-power enough, inexpensive and serves better than 32 kilobits per second, so that the whole thing is feasible. By the end of 2001, you'll see at least the first generation of those devices. Palm VII goes in this direction, but not far enough.

Margie:What do you now think the future holds for Linux and OE/ONE?

Eid: For Linux, consolidation in two or three major vendors who will dominate their respective market segments. Broader application/hardware support for Linux, which will make it a true alternative to Windows on the desktop. Corel's

work on the desktop and its rich set of applications will greatly help the Linux cause.

I expect OE/ONE to lead the way in the consumer electronics computing era. Our vision is to unify the way information, content and traditional applications are accessed and enjoyed. Our realistic goal is to eliminate the complexity from simple daily computing tasks. Using an information appliance should be as easy as turning on a TV and as fun as playing with a gaming console. OE/ONE will achieve this with the help of our current and future strategic partners.

Margie: Anything else you would like to tell us about that I haven't asked you?

Eid: We always want to remind people that we are, after all, a Linux company. So even though the market we are addressing is a consumer market, we want to hide the complexity of Linux as much as we can from the users. But if it weren't for open source, and Netscape Communicator coming to open source through Mozilla, you wouldn't see OE1 or companies like OE1 around today.

Margie: Thanks for your time. I am sure you will find success in your venture.

Archive Index Issue Table of Contents

Advanced search

Linux Means Business: A Case Study of Pakistan On-Line

Rafeeq Ur Rehman

Issue #72, April 2000

Linux is being used by an ISP in Pakistan—Mr. Rehman tells us why.

Pakistan On-Line (POL), http://www.pol.com.pk/, is an Internet Service Provider (ISP) operating in major cities of Pakistan. All of its Points of Presence (POPs) are operating on Slackware Linux, with some minor setup variations on each site. The ISP's local backbone uses fiber optic cable from a local telecom company, Pakistan Telecommunication Company Limited. Cisco routers are used at each POP to manage this backbone. POL has multiple links to the Internet and uses routing algorithms to manage traffic.

Other than Internet access services, the company also provides services for network design, installation, web development and hosting, domain name registration, etc. For remote-access services, POL uses Cisco and Xyplex access services at major POPs and Linux with Rocketport cards at smaller sites. Linux is used on HP Netservers and Compaq machines to provide Internet services like SMTP, POP, DNS, WWW, FTP, Proxy, etc. User authentication is done through a RADIUS server running on a Linux machine. User accounting and billing are done through RADIUS for Cisco remote access servers and Syslogd where Xyplex remote access servers are used. The accounting and billing process is also carried out on Linux through custom software developed in C.

Why Linux?

Choosing a network operating system for Internet operations is a very important decision. You need to be certain about the stability of your entire system, as the setup has to serve so many clients around the clock. Nobody can afford a service outage, even for a few minutes. Since we planned to use Intel-based servers, we had to choose between a Windows NT server and Linux. The most important factor was not the cost of the operating system, but the stability of the ISP facility. Our criteria for selecting an operating system depended on the following factors.

Previous experience: we had a very good experience with Linux, as we had already built large Linux-based networks. Some of these are being utililized in a similar environment, and some in educational institutions.

Stability: Linux was well-tested for stability by our staff in operations of different nature for many years. We were quite confident that it would not give us any problems in our operations.

Cost: although the cost of the operating system itself is not a major factor, when you add the other utilities and software required for an ISP, it is something to be considered. Not only is Linux free of cost, but you can also find all the required software for an ISP entirely free on the Internet. This includes mail servers, web servers, FTP and DNS servers that come bundled with Linux, making it a complete Internet solution. Additionally, if any of the available software does not serve a particular purpose, you can easily try one of the many available alternatives. For example, if you feel Sendmail is too complex to administer and have a mixture of UUCP and SMTP services, you can use Smail instead, which is quite easy to administer and a very useful UUCP-to-SMTP gateway.

Support

Ease of administration, customization and support on the Internet is another major issue. If you want support for commercial software, you have to pay someone on a regular basis. Linux is perhaps the only product for which you can get on-line help around the clock from so many experts all over the world without spending a single penny. It is quite fun to go to an IRC channel and join the discussion.

Tools Available with Linux

Although commercial operating system vendors are trying to bundle everything with their products, it is simply impossible for any of these to provide a number of utilities comparable to those available on Linux. People have built many tools and utilities for ISP operations in particular. For example, you can find many tools for analyzing web traffic logs, monitor the utilization of your Internet bandwidth, manage user accounts, check security holes, etc.

Linux Mail Servers

There is simply no match for the mail servers available on Linux. Using Sendmail, qmail or Smail, you can do anything you wish. If you want to go for ease of use in a mixed UUCP and SMTP environment, use Smail 3.2. If you want a complete, thoroughly tested, comprehensive solution, use Sendmail. If you

just want to give support for SMTP, IMAP, virtual domains, etc., use qmail and so on. You are free to make your choices depending upon your environment.

Web Server and Virtual Domains

At POL, we have been using Apache from the beginning. We have found the Apache web server excellent where stability and performance is concerned. We have many virtual domains running on a single web server, utilizing only one IP address. Basic configuration and virtual hosting for Apache is quite simple. You can also use SSL, URL caching, and protected user directories with Apache. All of the POL users have their own web pages which are kept in their home directories on the Apache server, and are free to update these pages any time they wish.

The Apache server is of great commercial use, as you can utilize it with databases, ODBC, etc. In fact, we have tested it with mSQL, ODBC and MySQL at Pakistan On-Line. We have found some clients who are interested in co-locating their database server at the POL premises and linking them to the Apache server for on-line databases. We are sure this will be a great success for us here in Pakistan.

Firewalls, Masquerading, Transparent Proxying

Pakistan On-Line is using the Linux kernel features for firewalling, IP masquerading and transparent proxying. We have installed Linux-based firewalls for some of our corporate clients. We have tested both packet filtering and proxy-based firewalls. For a proxy-based firewall, we have used Trusted Information Systems (TIS) Firewall Toolkit (FWTK), which is freely available in source code form on the Internet.

IP masquerading has been useful in environments where IP addresses are very sparse. In fact, this situation prevails in most of the developing countries where we do not have enough IP addresses for corporate clients. IP masquerade plays a very important role in those circumstances where we can use a Linux box to support a virtually unlimited number of computers to connect to the Internet through a single legal IP address.

Some of POL's corporate clients have dozens of computers on their private networks where they need Internet access. It is difficult for us to assign as many IPs to these customers as they want. We use the Linux IP masquerading feature on a computer running Linux, which then acts as gateway for the private LAN. Now the company is free to add as many computers on their private LAN as they want, without consulting the ISP again and again.

This arrangement has been very useful for us and has given POL an edge over other ISPs here. Other ISPs try to provide such a solution based on the Windows NT Server and Microsoft Proxy Server, which are costly for both hardware and software. Also, the NT-based system does not pass through all protocols as transparently as Linux does.

The transparent proxy feature in the Linux kernel is also very useful. We are using the transparent proxy feature with the help of Cisco routers to force all WWW traffic to pass through our proxy server. On the Cisco router, we have extended access lists which redirect all outgoing traffic on port 80 to pass through the Linux server. Another package, **tproxyd**, has also proven to be very useful in our operations. More information on tproxyd can be obtained from its web site (see Resources). A sample Cisco access list that serves the job might look like this:

```
interface Ethernet0
 ip address 209.58.13.1 255.255.255.0
 ip policy route-map proxy-redir
!
access-list 101 permit tcp 209.58.13.0 0.0.0.255 any eq www
route-map proxy-redir permit 20
 match ip address 101
 set ip default next-hop 209.58.13.6
!
```

Now, when the router receives an IP packet with a destination port equal to 80 from any computer on local network 209.58.13.0/24, it redirects this packet to 209.58.13.6, which is a Linux server running as the proxy server. The Linux **ipfwadm** utility is then used to manage this kind of traffic.

As far as caching is concerned, **squid** has remained our choice. It provides very good performance as well as stability. This is also used with the transparent proxy feature of Linux to obtain extra benefit. A number of tools for squid which analyze performance and scan logs in graphical format are available on the Internet.

Accounting and Billing

The ISP accounting and billing system is the most important thing, because this is the process that generates money for an ISP. It needs to be very accurate and stable. We have three types of systems that support dial-in users:

- Linux-based servers with multiport cards. The login and logout information from these is obtained through **syslogd**.
- Xyplex Max 1640 servers that can send both RADIUS and syslogd information. We are using syslogd information for billing purposes.
- Cisco remote access servers, which are sending accounting information to the RADIUS server.

We have developed a billing system in C that gets information from all three types of servers and generates user log files. Any user can see his billing information whenever required. We also use shell scripts for some housekeeping jobs in our billing system. It has proven to be a very good and user-friendly system, and two other small ISPs now use this same billing system.

Low-Cost Access Server with Linux and Rocketport

When you plan a very small ISP site or corporate network with a small number of dial-in users, it may be useful to deploy Linux as a remote access server. This saves a lot of money. We are using Linux with the Comtrol Rocketport multiport PCI adapters for this purpose. One Linux system can support up to four adapters, each with 32 high-speed serial ports. Thus, in total you can have 128 ports available in a single Pentium-based computer with good throughput. Since Rocketport cards have on-board intelligent processors, these do not pose much load on the computer. Now, just by making some changes in the login procedure, you can configure many things. For example, you can set the number of simultaneous logins you will allow for a single login name.

Conclusions

Linux is a very useful and stable operating system for ISP services. It provides a cost-effective, user-friendly, easy-to-configure environment. The number of utilities available for ISP operation and support is excellent. The biggest advantage is that it can be used in almost any environment and provides an edge over your competitors.

Resourses

Rafeeq ur Rehman (rehman@pol.com.pk) received bachelor's and master's degrees in Electrical and Computer Engineering, respectively, from the University of Engineering and Technology, Lahore, Pakistan. His main areas of interest are computer networks and distributed computing. He has been using Linux since kernel version 0.0.99.

Archive Index Issue Table of Contents

Advanced search

Building Your Own Internet Site

Tony Dean

Issue #72, April 2000

A quick look at what you need to build a web site for your personal or business needs, with pointers to the details.

When I needed to build a development and demonstration system to start my own Internet-based business, I decided to explore how to build an Internet site for education and low-end business support. The technologies I chose allow an individual to build a site he or she can control and use for experimentation.

If you own your own site, you have complete freedom to explore technologies like Internet security, CGI (Common Gateway Interface) and Java servlet development. Much of the work in this area is handled behind the doors of an ISP (Internet Service Provider); there are significant costs involved in CGI and other server-side programming.

These technologies are difficult to deal with, since they may introduce security flaws to an Internet site. Trying to stress the security would certainly stress the system administrator. Features like CGI and Java Servlets are programs that attach to the back side of a web server. The owner of the web server would be negligent if he simply allowed a customer to add programs without serious testing.

When you are using an ISP, you are limited by the ISP to specific technologies. Many ISPs will not permit CGI and Java servlet programs to be used with even their basic ISP plans. More advanced business plans often allow CGI or Java servlets, but will restrict upgrades to your site by requiring that you pay them to test your code before it goes live. This introduces delay, and control limitations that may prohibit some features from being deployed on your site.

To the small business, a personal Internet site offers flexibility that is usually the domain of larger competitors. When you own the site, you can make

changes quickly and at relatively low cost. You must also bear the responsibility for maintaining a secure site and liability for the content of the site.

To keep the cost of our proposed “Personal Internet Site” low, the majority of the software used will be freely available operating system, programming tools and network software. The Free Software Foundation and thousands of developers around the world have contributed to a large base of freely available software. The availability of free or nearly free software makes constructing the system outlined in this article an achievable goal.

Scope of the Initial System

The initial system will consist of several Intel-based personal computers running Linux. For this project, I have selected SuSE Linux 5.3. This is a recent distribution, with support for most of the features needed for an Internet site. The system will support a web server with a Java servlet runtime module. This configuration supports dynamically created web pages using a JDBC (Java Database Connectivity) compliant database.

Three PC class systems (AMD 5x86 133) and two K6-2 systems at 300+ MHz will constitute the processing core of this site. One system will handle firewall, proxy and routing duties. (For details on setting up these services, see “My Linux Home Network” by Preston Crow in this issue.) These are the services that allow our site to be visible to the Net and able to see the Net safely. The other systems will support production services and development support, respectively. Possibly an old 486 DX2 66 will be dusted off for a very light Internet appliance task.

An appropriate modem supporting the physical and link-level connectivity will be installed on the slowest system that can keep up with the data stream while handling the task of firewall chores, DNS and routing. By trimming the OS down to minimal-required functionality, it can be small enough to allow an older machine to be used for this dedicated connectivity task. The Linux Router Project (LRP) has a minimal configuration defined to support systems as small as a 386SX 16 with 8MB of RAM. (For details, see “The Linux Router Project” by David Cinege, *LJ*, March 1999.) This is an input- and output-intensive process, and it doesn't take much computer to keep up with a 128KBps data stream.

The Linux OS with networking support for IP configuration will be standard on all systems. In order to best optimize the OS, I have reconfigured and compiled the Linux kernel using the parameters required to provide networking features for an Internet site. This includes all support for the hardware network interfaces as well as IP networking and firewall options. All nonessential capabilities, such as multimedia features, are removed.

Web support consisting of the Apache web server with Java servlet extensions is already installed and running. Early prototyping of production systems will include Java JDBC server-side support. (See “Using Java Servlets with Database Connectivity” by Bruce McDonald, *LJ*, June 1999.) Database chores will initially be handled by PostgreSQL. (See “PostgreSQL—The Linux of Databases” by Rolf Herzog, *LJ*, February 1998.)

Fast Ethernet is used between the existing systems at the site. Systems not visible to the Internet will use IP (Internet Protocol) masquerading to enhance their security. (See “Setting Up a Linux Gateway” by Lawrence Teo in this issue.)

Required Services

Connecting to the Net requires that we deal with issues related to physical and logical connectivity. The physical connectivity issues are based on the selection and provisioning of telco (telephone company) or cable lines. The options for physical connectivity with modest bandwidth appear to be ISDN (Integrated Services Digital Network), DSL (Digital Subscriber Line) and cable IP services.

The services I need to connect to the Internet fall into two categories: telco services and IP access provider. I really don't need ISP (Internet Service Provider) services, because I will be my own ISP. All the services an ISP usually provides are available on the system as I have it configured.

Of the technologies available, ISDN is the oldest, dating back to 1978. DSL and cable modems are relatively new technologies and availability is limited for all these technologies. (See “The (not so) Wonderful World of High-Speed Internet Access” by Jason Schumaker in this issue.) Since DSL and cable IP are not available in my area, connectivity will be provided by an ISDN line. Internet access is provided by US West and can be connected 24x7 (24 hours a day, 7 days a week) with “on demand” service. In the interest of keeping the initial cost of the site down, I have arranged to obtain a used 3Com Impact IQ ISDN network adapter (they really aren't modems) to attach my system to my selected IP access provider.

On the logical side of connectivity, we need routing support and an Internet address. In this case, the US West Network will provide routing and DNS (Domain Name Services) support to my site. To be addressable, I need an IP number for my site. Eight static IP numbers are being rented from US West for $15.00 US per month. This yields five usable IP addresses, with the remainder used for configuration purposes. (See “Simplified IP Addressing” by Gene E. Hector, *LJ*, January 2000.)

Becoming Visible

This configuration has been successfully tested from the Net. I connected the system, noted its dynamic address, then logged in to the system with HTTP, TELNET and FTP sessions from a remote location. This assures me the system configuration is ready to support the services I plan to use for this Internet site.

To be visible to users of the Internet, the site will need to have a domain name that is registered with an organization called Internic. The web page at http://www.internic.net/ has the information and forms to register a domain name. In order to register for a domain, you must have a primary and a secondary DNS with which you define your system. The people who are responsible for those systems should be aware of your intentions, or you may find your site falling off the Net.

A domain name is required to be visible as something other than its assigned IP number. It would be difficult to remember network addresses in the IP number format of ###.###.###.### (e.g., 192.168.1.1), so a domain name puts our site in the form of "mySite.com". The Internet Access Provider will have to provide DNS visibility to the rest of the Net for my five static IP addresses and the domain name or names they represent.

Once I have a domain name, I can add aliases and extensions to define additional systems and services. For web access, a simple http://www.mySite.com/ produces a URL to access web pages. E-mail provided by an ISP is usually rather limited, often to five or fewer mailboxes. When you own the site and the mail router, you can have as many mailboxes as you wish. An outside user can send mail to me@mySite.com, and my own mail server will get it to my mailbox.

Conclusion

It is possible to start up an Internet site with typical ISP services in your home office. The availability of industrial-grade software for free is one of the key elements that makes this possible. Low-cost computers would not be enough if you had to add tens of thousands of dollars in license fees to the system. This quickly becomes prohibitive without free software.

From an educational perspective, this is an excellent platform for expanding skills. The OS and tools provide a high-quality system, with source code to delve into as necessary. Open standards are strongly supported and all major Internet development languages are available.

A variety of skills are related to developing and managing an Internet presence. This configuration can be used to study Internet site security, including

common tools like Satan or Tripwire. These two tools help an administrator verify security and help detect breaks in activity, respectively. Other uses of the site once it's up include e-commerce and application server development. Using technologies like JDBC (Java Database Connectivity) and CORBA allow the development of significant commercial projects.

Of course, if you should outgrow these systems, it is possible to move up to RISC-based hardware with Linux, as it runs on DEC Alpha, PowerPC, SPARC and MIPS processors as well as Intel. The upgrade path to other hardware and other UNIX implementations is much easier from Linux than from an Apple, a Windows NT Server or proprietary network elements.

email: tdean@du.edu

Tony Dean can be reached via e-mail at tdean@du.edu.

Archive Index Issue Table of Contents

Advanced search

The (not so) Wonderful World of High-Speed Internet Access

Jason Schumaker

Issue #72, April 2000

Discover the joys of using the new technology of DSL and cable modems to make your link to the world.

The idea seemed simple enough: I wanted to get a high-speed Internet connection at home, preferably using DSL (digital subscriber line). Thinking a quick phone call and a few hundred bucks in connection/installation fees would do the trick, I found instead that the business is still in its infancy and mired in a not-so-nice competition between telephone companies (telcos), DSL providers, cable providers and ISPs. There is a lot of money to be made and monopoly theories already abound. These have brought in the FCC, special-interest groups and others. It really is a mess. This is cutting-edge technology that may soon connect a majority of computer users to the Internet, and yet again, big businesses are too concerned with private interests. Legal wrangling will continue to delay the expansion of service and lead to much frustration for those waiting.

What are the Options?

When I began to research alternatives for high-speed access, I felt DSL service was the best solution. We use it here at *Linux Journal* and the speed of the connection is wonderful. For no particular reason, I had a negative opinion toward a similar service offered through cable companies. I was surprised by what I found. As it turns out, cable has advanced quite a bit more than DSL service—there are over two million subscribers to cable, and only some 500,000 to DSL services nationwide (www.usatoday.com/life/cyber/tech/ctg704.htm). Both DSL service and cable are viable options.

What is DSL?

DSL is technology that uses ordinary copper phone wires to transmit high-bandwidth information to homes and small businesses. Your phone company receives information in analog and digital form. Digital information is converted to analog (voice) information and sent to your computer. Your modem converts the analog information back to digital. With DSL technology, the phone company can send digital information directly to your computer as digital data (you need a DSL modem, actually, a router), which allows for a more efficient use of bandwidth. Most importantly, the signals are separated, allowing you to talk on the phone and surf the Web at the same time.

Speed

The main advantages of DSL are speed and always-on Internet access. Having used DSL service here at work, I no longer have the patience for "old" dial-up modems. On Whatis.com, I found the statement "Using DSL, up to 6.1 megabits per second of data can be sent downstream and up to 640Kbps upstream." This will dramatically improve the speed of your Internet connection—as much as 25 times faster than a 56k modem, according to US West—and without all the horrible squeaks and scratches of a modem. With a DSL line, you can quietly and quickly zip through the Internet, download images or just surf at your leisure.

The service I could afford would provide a 384Kbps downstream connection and a 128Kbps upstream connection. This is called ADSL (asymmetric), since different speeds are used for the downstream connection (the data sent to the user) and the upstream connection (the user requests). Requesting data uses little bandwidth, but the transmission of video, audio and 3-D images requires more. Splitting bandwidth in this manner allows for a small portion of the downstream connection to be used for phone conversations. Imagine being able to talk on the phone to a friend while checking out the same web site, or sending a fax without having to disconnect from the Net!

Requirements and Cost

On average, DSL service runs between $30 and $60 per month. The initial installation and activation fees can be $300 or more, but special offers abound. The service I chose will cost $60 per month and has an eight-week waiting period before installation. As far as hardware is concerned, I need a router and an Ethernet card. You can spend close to $250 on a router and another $50 for the Ethernet card. If you are technically inclined, handling the installation of the router and Ethernet card yourself can save you as much as $200. The router or "DSL modem" runs about $250, but again, rebates and special offers seem

common. My initial fees after all the rebates, etc. would be around $250, which seems reasonable.

Frustration comes from having to pay $60 per month. Our Editor in Chief, Marjorie Richardson, lives a mere 20 blocks from me, but falls within US West coverage. She pays $30 per month, and US West supplied the router and Ethernet card (initial fee $140). I pay more because I am not within US West coverage, and therefore subscribe through the competition. I would go through Covad Communications, who leases the lines from US West. Then Covad recommends an ISP (oz.net), and I get a headache—three companies to deal with! This is very common and leads to customer service problems. If I have a problem with my service, I would first call my ISP who might say the problem is with the DSL provider (in my case, Covad), who might then point a finger at US West in an endless round of blame-placing, rather than trying to find a solution.

The G.lite Standard

It has been more than a year since the International Telecommunication Union (ITU) voted in favor of making G.lite ADSL (or ADSL Lite) standard (www.pcworld.com/pcwtoday/article/0,1510,8546,00.html). This *PC World* article states, "The G.Lite modem standard, called G.992.2, uses a stripped-down version of asymmetric digital subscriber line." The connection is a bit slower than full-rate ADSL, but, according to ADSL Forum (http://adsl.com/), "the full-rate standard is more costly and problematic. Installation requires the phone company to come out and install a splitter on your phone line." With the G.lite modem, this feature is installed into the modem, leaving only the installation of the modem to the customer. This keeps costs down for both customers and telcos, which is why many in the industry expect the G.lite modem to prosper.

And Then There was Cable

Cable is currently more popular and more readily available than DSL service. Currently, cable modem services are capable of reaching nearly 52 percent of U.S. households, compared to just 23 percent for telephone companies (John Borland, CNET News.com, 1/13/00). This trend is expected to continue for a few years, but by 2003, twice as many DSL installations as cable are forecast. The cited reason is phone lines are more available, especially in business districts ("Cable vs. DSL", *Seattle Times*, May 2, 1999). There are those who would disagree, believing cable's current market position will only increase. AOL's recent acquisition of Time Warner could be evidence enough to support such claims. With access to Time Warner's cable lines, many believe AOL will place a higher priority on cable than DSL. Regardless, phone lines reach further into rural areas, and currently provide a more consistent service in higher-populated areas.

Cable service is supposed to be faster than DSL (by up to 10MBps), but since users are clumped together in hubs, speed can be greatly affected as the number of users rises. The cost of cable is slightly lower, as are installation fees. Prices for TCI's @Home cable Internet service is currently $39.95 per month, with a $150 installation fee. However, you do not have the option of choosing an ISP—that is part of the package, and not surprisingly has led to a few lawsuits. The basics are that smaller ISPs want access to cable lines, and the larger companies don't want to give it. As it stands now, AT&T has one ISP it uses—Excite@Home. Another ISP wanting to use AT&T lines would be charged for using those lines, which would mean the customer cost would go up accordingly.

I should note that I could find no indication that the TCI cable service supports Linux, while all those involved in my potential DSL plan do. And the TCI@Home web site says it "is for residential, casual use only and does not support servers."

Jockeying for Position

Bringing costs down will have to become a major priority for both cable companies and the telcos. In an interesting move, on December 6 AT&T decided to offer the lines to competitors, although terms are not yet clear. What is clear is that many are fighting for entry into what promises to be a lucrative industry. A few ISP's are attempting to offer free DSL service by as early as March 15. Broadband Digital Group (a southern California company), iNYC (New York) and Staruni (Beverly Hills) are currently working to do just that (John Borland, CNET News.com, 1/13/00). The ISPs will collect demographic information by requiring users to click on a certain number of banner ads daily. They will then, no doubt, turn around and sell their findings to advertisers, who will complete the circle with more banner ads. Analysts are skeptical of this plan, since the cost of providing high-speed Internet access will still be high.

The FCC to the Rescue?

Covad extends DSL coverage but has to buy lines from US West, resulting in an extra $20 to $30 charge per month. Basically, I am forced to split my phone line into two lines (which is exactly what I had hoped to avoid) and am being charged accordingly. This situation may soon change. An article in *ZDNet*, "FCC Mandates Line Sharing" from November 18th, 1999, states, "The Federal Communications Commission today ruled that regional telephone companies must share local-loop lines with Digital Subscriber Line competitors." Of course, the telcos are less than happy about this, as they are reluctant to undercut their current T1 services. Regardless, it seems the result will be cheaper DSL subscriptions, which should ensure the propagation of DSL technology. The

mandate will be challenged, and people like me will just have to deal with inflated prices until the legal wrestling is finished.

Do I Need High-Speed Internet Access?

I have decided I am not ready to be a part of this tangled web just yet, as I don't want to pay the extra fees. While I do not possess the patience of Job, I can manage with a 56K modem for now, at least until the big corporations hammer out accords and mergers, etc. I will stress that I *want* DSL service. Once you have tried it, you will want it, too. Why run when you can drive? Why cook with an oven when there's a microwave? The speed achieved by using DSL or cable modem makes browsing the Internet fun again.

Whether or not you make the plunge depends on what you wish to accomplish and how much patience you have. If you run a business, be it from home or not, a super-fast Internet connection makes sense. Overall, the service is more of a *want* than a *need*, but that will change. We are a society hungry for speed and technology. Dial-up modems have served a purpose and will now give way to innovation. Within the next five years, you will most likely be receiving Internet access, phone service, long-distance service and cable from one company. In December 1999, Bell Atlantic was cleared by the FCC to offer long-distance telephone service. This is the first time, since the breakup of AT&T, that this has been allowed. What it means for Bell Atlantic is the ability to extend coverage. If other telcos are given the same clearance and DSL technology becomes widespread, thoroughly understood, and fully supported, prices will drop and the traditional dial-up modems will become door stops.

Jason Schumaker is an Assistant Editor at *Linux Journal*. He can be reached at info@linuxjournal.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

RTAI: Real-Time Application Interface

P. Mantegazza

E. Bianchi

L. Dozio

S. Papacharalambous

S. Hughes

D. Beal

Issue #72, April 2000

An introduction to RTAI for deterministic and preemptive real-time behaviour for Linux.

Before jumping headlong into our discussion of RTAI, a quick review of what real time means and how standard Linux relates to real time is appropriate. The term “real time” can have significantly different meanings, depending on the audience and the nature of the application in question. However, computer science literature generally defines only two types: soft and hard real-time systems.

A “soft real-time system” is characterized by the ability to perform a task, which on average, is executed according to the desired schedule. A video display is a good example, where the loss of an occasional frame will not cause any perceived system degradation, providing the average case performance remains acceptable. Although techniques such as interpolation can be used to compensate for missing frames, the system remains a soft real-time system, because the actual data was missed, and the interpolated frame represents derived rather than actual data.

“Hard real-time systems” embody guaranteed timing, cannot miss timing deadlines and must have bounded latencies. Since deadlines can never be missed, a hard real-time system cannot use average case performance to

compensate for worst-case performance. One example of a hard real-time task would be monitoring transducers in a nuclear reactor, which must use a complex digital control system in order to avoid disaster. RTAI provides these necessary hard real-time extensions to Linux, thus enabling it to be targeted at applications where the timing deadlines cannot be missed.

Issues with Real-time Support in the Linux Kernel

RTAI provides hard real-time extensions to Linux, yet standard Linux has support for the POSIX 1003.13 real-time extensions, so how does the "real-time" capability of standard Linux fall short?

The POSIX 1003.13 standard defines a "Multi-User Real-Time System Profile" which allows for "real-time" processes to be locked into memory to prevent the process from being paged to hard disk, and a special scheduler which ensures these processes are always executed in a predictable order.

Linux meets this standard by providing POSIX-compliant memory lock (mlock), special schedule (sched_setsched) system calls and POSIX RT signals. While those features do provide improved deterministic performance, the resultant tasks cannot be defined as hard real-time tasks, since the soft real-time process can be blocked by kernel activity.

The standard Linux real-time POSIX API and the real-time kernel offer dramatically distinct services, especially within a multi-user and multi-tasking application. A simple program in pseudo-code, shown in Listing 1, can be used to demonstrate the system performance and issues surrounding the POSIX approach.

Listing 1

This program reads the absolute time before entering a system call which is used to suspend the process for a predetermined interval. Next, the program reads the absolute time after the system call returns. If the system is heavily loaded with a high level of kernel activity, the return from the system call will be delayed by that activity, thus the magnitude of this delay is defined by the difference between the expected sleep time (i.e., 100 milliseconds) and the actual sleep time.

For a standard Linux system that is idle, the above task performance is extremely stable and the timing deviation is very low, yielding a worst-case deviation from an average of approximately 100 microseconds on a Pentium II 300MHz system.

However, if the program is run on a standard Linux system, using the same POSIX approach but with simultaneous I/O activity, the performance dramatically deteriorates, since the facilities of standard Linux do not allow any way for the test program to preempt the I/O-intensive application. The average execution quickly becomes unacceptable when, for example, one edits a very large file and causes hard disk activity at the same time the test program is running. This inability to preempt causes the test program to suffer typical deviations of 30 milliseconds, and for the worst case, in excess of hundreds of milliseconds. Consequently, standard Linux processes using the POSIX approach cannot be used for reliable hard real-time performance in multi-tasking environments.

Although members of the Linux community have discussed introducing a low-latency scheduler in future kernels to enhance soft real-time performance, standard Linux will still fall well short of the guaranteed response time required by a hard real-time task.

The performance advantage of real-time Linux is easily seen by running this same test program as a real-time task, yielding worst case deviations of less than 12.5 microseconds regardless of I/O activity.

Along with the scheduling benefits of the real-time kernel, the efficiency of the real-time architecture allows Linux to run under aggressive task iteration rates. For example, when running this program at 500-microsecond intervals as a POSIX real-time task, Linux stops completely. Running this program as a real-time task leaves enough resources for Linux to continue to run.

The operating requirements of a general-purpose operating system such as Linux are different from those of a hard real-time system. This difference leads to other issues, summarized below, which limit the potential for standard Linux to act as a real-time operating system.

- The Linux kernel uses coarse-grained synchronization, which allows a kernel task exclusive access to some data for long periods. This delays the execution of any POSIX real-time task that needs access to that same data.
- Linux does not preempt the execution of any task during system calls. If a low-priority process is in the middle of a **fork** system call and a message is received for the video display process, then the message will unfortunately be held in the queue until the call completes, despite its low priority. The solution is to add preemption points in the kernel, having the deleterious effect of slowing down all system calls.
- Linux makes high-priority tasks wait for low-priority tasks to release resources. For example, if any process allocates the last network buffer

and a higher-priority process needs a network buffer to send a message, the higher-priority process must wait until some other process releases a network buffer before it can send its message.

- The Linux scheduling algorithm will sometimes give the most unimportant and nicest process a time slice, even in circumstances where a higher-priority process is executable. This is an artifact of a general-purpose operating system that ensures a background maintenance process; e.g., one that cleans up log files and runs even if a higher-priority process were able to use all the available processor time.
- Linux reorders requests from multiple processes to use the hardware more efficiently. For example, hard-disk block reads from a lower-priority process may be given precedence over read requests from a higher-priority process in order to minimize disk-head movement or improve the chances of error recovery.
- Linux will batch operations to use the hardware more efficiently. For example, instead of freeing one page at a time when memory gets tight, Linux will run through the list of pages, clearing out as many as possible, which will delay the execution of all processes. This is clearly desirable for a general-purpose operating system, but is undesirable for real-time processes.

Real-time and general-purpose operating systems have contradictory design requirements. A desirable effect in one system is often detrimental in the other.

Unfortunately, any attempt to satisfy both requirements in the same kernel often results in a system that does neither very well. That is not to say general-purpose functionality and real-time determinism cannot be achieved simultaneously. In fact, operating systems that combine these requirements exist now, and they are indeed deterministic, preemptive and contain bounded latencies (thus meeting the requirement for a hard real-time system). However, the worst-case behavior for those latencies can be unacceptably slow.

An Introduction to RTAI

In order to make Linux usable for hard real-time applications, members of the Department of Aerospace Engineering, Politecnico di Milano (DIAPM) envisioned a real-time hardware abstraction layer (RTHAL) onto which a real-time applications interface (RTAI) could be mounted. Unfortunately, further investigation revealed that the Linux kernel available in late 1996, 2.0.25, was not yet mature enough for the concept.

Around the same time, a group headed by Victor Yodaiken at the New Mexico Institute of Mining and Technology (NMT) in Soccorro, NM introduced its real-time Linux (RTLinux), which provided the DIAPM team with the opportunity to

learn further about the Linux kernel, the hardware and the modifications necessary to provide preemptive and deterministic scheduling. The turning point came in 1998 when the Linux 2.2.*x* kernel featured key improvements, including much-needed architectural changes to the Linux/hardware interface. These changes, combined with the experience gained by the DIAPM team while working with their own evolution of the NMT-RTLinux kernel, and the concepts of 1996, resulted in RTAI.

RTAI provides guaranteed, hard real-time scheduling, yet retains all of the features and services of standard Linux. Additionally, RTAI provides support for UP and SMP—with the ability to assign both tasks and IRQs to specific CPUs, x486 and Pentiums, simultaneous one-shot and periodic schedulers, both inter-Linux and intra-Linux shared memory, POSIX compatibility, FPU support, inter-task synchronization, semaphores, mutexes, message queues, RPCs, mailboxes, the ability to use RTAI system calls from within standard user space and more.

RTAI Architecture

The underlying architecture for RTLinux and RTAI is quite similar. For each implementation, the Linux operating system is run as the lowest priority task of a small real-time operating system. Thus, Linux undergoes no changes to its operation from the standpoint of the user or the Linux kernel, except that it is permitted to execute only when there are no real-time tasks executing. Functionally, both architectures provide the capability of running special real-time tasks and interrupt handlers that execute whenever needed, regardless of what other tasks Linux may be performing. Both implementations extend the standard Linux programming environment to real-time problems by allowing real-time tasks and interrupt handlers to communicate with ordinary Linux processes through a device interface or shared memory.

The primary architectural difference between the two implementations is in how these real-time features are added to Linux. Both RTLinux and RTAI take advantage of Linux's loadable kernel modules when implementing real-time services. However, one of the key differences between the two is how these changes, to add the real-time extensions, are applied to the standard Linux kernel.

RTLinux applies most changes directly to the kernel source files, resulting in modifications and additions to numerous Linux kernel source files. Hence, it increases the intrusion on the Linux kernel source files, which can then result in increased code maintenance. It also makes tracking kernel updates/changes and finding bugs far more difficult.

RTAI limits the changes to the standard Linux kernel by adding a hardware abstraction layer (HAL) comprised of a structure of pointers to the interrupt

vectors, and the interrupt enable/disable functions. The HAL is implemented by modifying fewer than 20 lines of existing code, and by adding about 50 lines of new code. This approach minimizes the intrusion on the standard Linux kernel and localizes the interrupt handling and emulation code, which is a far more elegant approach. Another advantage of the HAL technique is that it is possible to revert Linux to standard operation by changing the pointers in the RTHAL structure back to the original ones. This has proven quite useful when real-time operation is inactive or when trying to isolate obscure bugs.

Many have surmised that the HAL could cause unacceptable delays and latencies through the real-time tasking path. In fact, the HAL's impact on the kernel's performance is negligible, reflecting highly on the maturity and design of the Linux kernel and on those who contributed to its development.

Real-Time Application Interface

The HAL supports five core loadable modules which provide the desired on-demand, real-time capability. These modules include **rtai,** which provides the basic rtai framework; **rtai_sched,** which provides periodic or one-shot scheduling; **rtai_mups,** which provides simultaneous one-shot and periodic schedulers or two periodic schedulers, each having different base clocks; **rtai_shm,** which allows memory sharing inter-Linux, between real-time tasks and Linux processes, and also intra-Linux as a replacement for the UNIX V IPC; and **rtai_fifos,** which is an adaptation of the NMT RTLinux FIFOs (file in, file out).

Like all kernel modules, these can be loaded and unloaded (using the standard Linux **insmod** and **rmmod** commands) as their respective capabilities are required or released. For example, if you want to install only interrupt handlers, you have to load only the rtai module. If you also want to communicate with standard Linux processes using FIFOs, then you would load the rtai and rtai_fifos modules. This modular and non-intrusive architecture allows FIFOs to run on any queue and use immediate wake-ups as necessary.

The Real-Time Task

The real-time task is implemented similarly to RTAI; i.e., it is written and compiled as a kernel module which is loaded into the kernel after the required RTAI core module(s) has been loaded. This architecture yields a simple and easily maintained system that allows dynamic insertion of the desired real-time capabilities and tasks. The example below shows all that is required for a task to be scheduled in real time:

```
insmod /home/rtai/rtai
insmod /home/rtai/modules/rtai_fifo
insmod /home/rtai/modules/rtai_sched
insmod /path/rt_process
```

To stop your application and remove the RTAI:

```
rmmod rt_process
rmmod rtai_sched
rmmod rtai_fifo
rmmod rtai
```

The facilities **ldmod** and **remod** may be used to load and unload the core modules.

Task Scheduler

RTAI's task scheduler allows hard real-time, fully preemptive scheduling based on a fixed-priority scheme. All schedules can be managed by timing functions and real-time events such as semaphore acquisition, clocks and timing functions, asynchronous event handlers, and include inter-task synchronization.

One-Shot and Periodic Scheduling

RTAI supports both one-shot and periodic timers on Pentium and 486-class CPUs. Although both periodic and one-shot timers are supported, they may not be instantiated simultaneously; i.e., one-shot and periodic tasks may not be loaded into the kernel as modules at the same time.

Using these timers (instantiated by rtai_sched), periodic rates in excess of 90KHz are supported, depending on the CPU, bus speed and chip set performance. On Pentium processors, one-shot task rates in excess of 30KHz are supported (Pentium II, 233 MHz), and on 486 machines, the one-shot implementation provides rates of about 10KHz, all while retaining enough spare CPU time to service the standard Linux kernel.

The limitation of rtai_sched to support simultaneously one-shot and periodic timers is mitigated by the MultiUniProcessor (MUP) real-time scheduler (rtai_mups), which provides the capability to use, simultaneously, both a periodic and a one-shot timer or two periodic timers with different periods, at a performance equivalent to that noted above under rtai_sched. Note that, since the MUP uses the APIC (advanced programmable interrupt controller) timer, it cannot operate under SMP and requires each MUP-scheduled task to be locked to a specific CPU (thus the MultiUniProcessor designation); however, the MUP retains all coordination and IPC services, so that no other capabilities are lost.

Floating-Point Operations

Floating-point operations within real-time tasks/ISRs (interrupt service routines) are possible, provided these tasks are marked, upon loading, as tasks which

require the FPU. This method provides real-time task access to the FPU while still allowing FPU access to standard Linux tasks.

Interrupt Handling

RTAI provides efficient and immediate access to the hardware by allowing, if one chooses, interaction directly with the low-level PC hardware, without first passing through the interrupt management layers of the standard Linux kernel.

The ability to individually assign specific IRQs to specific CPUs, as described in further detail below, allows immediate, responsive and guaranteed interface times to the hardware.

Inter-Process Communication

The term inter-process communication (IPC) describes different ways of message passing between active processes or tasks, and also describes numerous forms of synchronization for the data transfer.

Linux provides standard system V IPC in the form of shared memory, FIFOs, semaphores, mutexes, conditional variables and pipes which can be used by standard user processes to transfer and share data. Although these Linux IPC mechanisms are not available to do real-time tasks, RTAI provides an additional set of IPC mechanisms which include shared memory, message queues, real-time FIFOs, mutexes, semaphores and conditional variables. These are used to transfer and share data between tasks and processes in both the real-time and Linux user-space domains.

RTAI's remote procedure call (RPC) mechanism is similar in operation to QNX messages available to real-time tasks, and transfers either an unsigned integer or a pointer to the destination task(s).

The RTAI mailbox implementation provides the ability to send any message from user space to real-time tasks, real-time tasks to real-time tasks, and user tasks to user tasks (using LXRT) via any definable mailbox buffer size. Multiple senders and receivers are allowed, where each is serviced according to its priority.

proc Interface

From version 0.9, RTAI includes a proc interface which gives useful information on the current status of RTAI, including schedulers loaded; real-time tasks activity, priority and period; and FIFOs in use and their associated buffer sizes. The development of even more features is currently underway.

SMP Support

RTAI provides true support for symmetric multi-processing (SMP) architectures through its task and IRQ management.

By default, all tasks are assigned to run on any CPU (of a SMP platform). Each task, however, can be individually assigned to any CPU subset, or even to a single CPU. Additionally, it is possible to assign any real-time interrupt service to any specific CPU. Because the ability to force an interrupt to a specific CPU is not related to the SMP scheduler, RTAI retains the flexibility to perform these two operations independently.

These capabilities provide a method of statically optimizing the real-time application, if manual task distribution handles the task more efficiently than the automatic SMP load-distribution services of Linux.

Linux-RT (LXRT)

Since real-time Linux tasks are implemented as loadable modules, they are, for all practical purposes, an integral part of the kernel. As such, these tasks are not bounded by the memory protection services of Linux, and they have the ability to overwrite system-critical areas of memory, bringing the system to an untimely halt. This limitation has been a large frustration to those of us who have erred during real-time task development.

RTAI's LXRT solves this problem by allowing the development of real-time tasks, using all of RTAI's hard real-time system calls from within the memory-protected space of Linux and under a “firm” real-time service. When the developer is satisfied with a task's performance within LXRT, the task is simply recompiled as a module and inserted into the kernel (along with the associated modules which provide RTAI's real-time features) to transition from firm to hard real-time.

LXRT's firm real-time service, similar to that offered by the Kansas University Real-Time (KURT) patch, provides soft real-time combined with fine-grained task scheduling. Performance under LXRT is quite good, yielding latencies not much greater than for a standard Linux system call leading to a task switch. Although this is very valuable as a development tool, we should not lose sight of the fact that RTAI's firm real-time implementation can prove especially useful for those tasks which don't require hard real-time, but yet are not quite satisfied with the scheduling performance of standard Linux.

POSIX Compatibility API

RTAI implements a compliant subset of POSIX 1003.1.c through the use of a single loadable module. These calls support creation, deletion, attribute control and environment control for threads, mutexes and condition variables. The resultant POSIX support is similar to standard Linux threads, except that parent-child functions (which are not appropriate for real-time tasks, since all threads are considered to be part of a single process) and signal handling (which is currently in development) are not supported.

Typical Performance

RTAI is now competitive from both a cost and performance perspective with the commercial RTOS currently available.

Since the performance of any RTOS system is determined by the performance of the RTOS itself, the performance of the hardware on which it is running, and the test procedure used to acquire the data, absolute performance figures are very difficult to quantify, often making comparisons difficult between fundamentally similar RTOSes. However, the data below was measured on, and is representative of, typical Pentium II 233MHz and 486 platforms.

For these performance characterizations, an early version of the RTHAL module was demonstrated running a timer at 125KHz (Pentium II, 233MHz) while simultaneously servicing Linux, which was working under a heavy load. During this demonstration, the average and maximum jitters about the periodic timer were 0<\#181>s and 13<\#181>s, respectively. This performance, combined with additional tests, can be summarized in this way:

- Maximum periodic task iteration rate: 125KHz
- Typical sampling task rate: 10KHz (Pentium 100)
- Jitter at maximum task iteration rate: 0-13<\#181>s UP, 0-30<\#181>s SMP
- One-shot interrupt integration rate: 30KHz (Pentium-class CPU), 10KHz (486-class CPU)
- Context switching time: approximately 4<\#181>s

Future Developments

RTAI continues to grow and mature through the combined contributions of the DIAPM development team and those across the Internet who are encouraged by RTAI's flexible architecture, high performance and feature set. While this real-time Linux extension is very capable, stable and mature today, work is far from finished. The future holds many things, including:

- VxWorks compatibility libraries

- pSOS compatibility libraries
- Enhanced development tools
- Enhanced debug tools
- Real-time Ethernet functionality

Resources

The information provided here was produced with respect to the current performance and feature sets of RTAI and RealTime Linux. However, the RTAI and Zentropix home pages always contain the most up-to-date information.

P. Mantegazza works for Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano.

E. Bianchi works for Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano.

L. Dozioi works for Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano and can be reached at luca.bianchi@infementia.it.

S. Papacharalambous works for Zentropix Computing, LLC. (Note: Zentropix was acquired by Lineo on February 1, 2000.)

S. Hughes works for Zentropix Computing, LLC. (Note: Zentropix was acquired by Lineo on February 1, 2000.)

D. Beal works for Zentropix Computing, LLC and can be reached at daveb@zentropix.com. (Note: Zentropix was acquired by Lineo on February 1, 2000.)

Archive Index Issue Table of Contents

Advanced search

Novell Adopts OpenLDAP

Craig Knudsen

Issue #72, April 2000

What's new at Novell...

Lightweight Directory Access Protocol (LDAP) is a client-server protocol for obtaining directory-based information. It was originally developed at the University of Michigan as a method to access X.500 directory information over TCP/IP.

Today, LDAP servers are typically stand-alone applications rather than gateways to X.500 directories. This isn't a terribly surprising development, since X.500 can be difficult to implement and resource-intensive for X.500 clients.

LDAP is an Internet standard controlled by the IETF and is used in products by Microsoft, Sun, Oracle, Netscape and many others.

While the description of LDAP sounds fairly boring, it's an incredibly useful tool. Most users' first experience with LDAP will be looking up someone's e-mail address on a large LDAP server like Bigfoot. Check out your address book in Netscape, and you'll see that you can use various LDAP servers to search for people.

Novell

Founded in 1983 as a LAN specialist, Novell is now best known for NetWare and Novell Directory Services (NDS), and is present in over 80% of Fortune 500 companies.

Novell's major competitor in the commercial network directory market is Microsoft, which is planning to unveil their Active Directory as part of the Windows 2000 release.

In January, Novell announced the planned release of their LDAP Libraries for C Software Development Kit. The new SDK will allow developers to use the LDAP API to access Novell's NDS eDirectory. While this was possible before using third-party LDAP libraries, this is the first Novell developer library to support the LDAP API. The LDAP SDK will be released as part of the Novell Developer Kit (NDK) in March.

The SDK is based on the OpenLDAP Project's client library, written in C. Created to promote and develop commercial-grade open-source LDAP applications and tools, the OpenLDAP Project is coordinated by the OpenLDAP Foundation, a not-for-profit corporation.

How will this new relationship between Novell and OpenLDAP benefit OpenLDAP and the Open Source community? Novell is now a member of the OpenLDAP development community and is contributing enhancements, bug fixes, testing and documentation.

According to OpenLDAP's chief architect Kurt Zeilenga, “Novell developers are active on project mailing lists and have contributed a number of minor enhancements and bug fixes. We look forward to more significant contributions from Novell.”

The impact of Novell's contribution to the OpenLDAP project will not likely be seen by OpenLDAP users until the release of OpenLDAP 2.0, scheduled for general release sometime in the first quarter of 2000.

Not surprisingly, Novell's plans extend beyond just helping the OpenLDAP Project. This includes enhancing the OpenLDAP client libraries by adding NDS-specific extensions that will eventually be available as source code from Novell, creating tutorials and enhanced API documentation and providing commercial technical support. Novell will offer commercial support through its Developer Support program, ranging from free newsgroup access to paid support technicians.

Novell's customers also stand to benefit from the LDAP SDK. Kris Magnusson, open-source architect for Novell, believes, “Novell's LDAP SDK promotes an open standard, LDAP v3, as the preferred access mechanism for NDS eDirectory. Because developers will be able to use the LDAP v3 Internet standard to access NDS eDirectory, they can write to a single, open SDK.”

In order to use the Novell LDAP SDK, Windows developers will use an InstallShield-based self-extracting executable, while NetWare installation will take place through Novell Installation Services. Linux and UNIX users will install

source or binary packages distributed as “tarballs”, a common choice for open-source projects.

Using open-source products as part of commercial products is becoming increasingly popular, and OpenLDAP's software is a perfect fit. As part of their eServer product, Caldera Systems has also included OpenLDAP's software.

Novell's adoption of OpenLDAP's open-source SDK and commitment to contribute source code and documentation signal a change in corporate strategy. Novell's LDAP SDK plans provide clear benefits to both the Open Source community and Novell, resulting in a win-win solution.

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. 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 JOURNAL

Artist's Guide to the Desktop, Part 2

Michael J. Hammel

Issue #72, April 2000

In this episode, Mr. Hammel tells us all about Enlightenment 0.16.1.

It's a simple fact of the desktop world: artists, like all other users, want lots of screen space. Enlightenment gives it to them, when they want it, and on their terms. Enlightenment, usually referred to as E, allows users more control of their desktop than just about any other window manager. Where Window Maker and AfterStep are highly configurable, E is downright malleable, bending and twisting to the delight of its users.

Window managers today share some common threads. They usually allow some level of configuration for the Title Bar. Most have some form of pager that allows the user to switch between virtual desktops and, in some cases, multiple pages within each desktop. They all support icons and most have either some form of icon box or a way of specifying where icons should be displayed on the desktop. All have menus for accessing applications and managing the system as a whole.

E takes all this a bit further. Perhaps you've seen the fancy window borders for which E gained its fame. Those are the themes you've heard so much about, the personalization that the Windows world calls “skins”. What you may not have seen are the various ways you can jump from desktop to desktop. You can drag windows between any page on any desktop, using the fully configurable pagers (one pager for each desktop). You can also slide desktops around, like giant sheets of paper, so that the top sheet only partially obscures the one below it. None of the other window managers has this particular feature. Some might call it fluff, but having quick access to your desktops in a point-and-click manner is a real joy for heavy users like me.

In this article, I will look at the configuration and use of E from the perspective of an artist—someone who wants to define not only his own unique look for the desktop, but also how the desktop should work. Although key and mouse

bindings are completely configurable right from the desktop in E, I'm going to discuss configuration using the default bindings. I'll also use the default themes, just so you don't get too confused by my own unique world.

I will assume you are not afraid of compiling and installing from source-code distributions. E is a powerful system, but it's young and still in early development. You can find fairly recent versions of E with most recent Linux distributions, although you may specifically have to request to have it installed. Additionally, since E is young and in a constant state of being updated, you will want to be familiar with building and installing software if you plan on getting serious with E. Getting and installing E are discussed in the article "Enlightenment Basics", which can be found in this month's "Strictly On-Line".

Up and Running—The First Time

Once you have the E environment ready, you can configure your login account to use it. The X Window System is arguably the most configurable user environment ever created (second only to Martha Stewart's kitchen). There are many ways you can configure your window manager after it's running, but determining which window manager to run can often be confusing. Big-name distributors of Linux desperately try to hide the apparent complexities of configuration from the user so that the desktop appears automatically configured. Fortunately, try as they might, it's easy to find a way around this.

The key comes in two parts: **startx** (and its compatriot, **xinit**) and the $HOME/.Xclients configuration file. **startx** is a shell script with a history that goes back to the early days of X. It's a fairly simple script, created to make using the more complex xinit (which is used to start the X server and any initial applications) easier for the typical user. Developers tend to build complex systems, only to spend the rest of their lives adding front ends and wrappers around them to simplify them for "the typical user". **xinit** will exist on all Linux systems with X installed. **startx** should be on all systems, although a few distributors may remove it in favor of their own homegrown version. If you're using one of those, dump it. There are cases where keeping historical pieces of software available, even if more advanced pieces become available, is advantageous. This is one of those cases. You should be able to find both xinit and startx under /usr/X11R6/bin.

X Session

As long as you use startx/xinit to launch your X session, you can make use of the .Xclients configuration file to determine what X will start. **xinit** eventually reads this file, which should be located in your $HOME directory, starts the X server, then runs the commands specified in .Xclients. For example, you might

specify that two xterms, xclock and xcalc all start up along with E. Such a configuration might look like this:

```
xterm &
xterm -bg black -fg wheat -geom 80x30\
   -font 7x13 &
xclock &
xcalc &<\n> enlightenment
```

Notice that this file looks like an ordinary shell script, but you don't have to specify the shell to use (i.e., there is no **#!/bin/sh** line at the top of the file) because the standard shell, /bin/sh, is used. Each line is a command to run, and, with the exception of the window manager line, each command must be placed in the background using the ampersand. The last command is normally the window manager of choice. No matter which command is last, it shouldn't be placed in the background. If it were, xinit (via startx) would then see that the commands file (.Xclients) has completed. This would cause xinit to exit, bringing your X session to an end and returning you to the text console.

Create an Xclients file like the one in the previous example. If X is already running, which is likely for most users who accepted the default configurations provided by their Linux distributions, you can still use any editor (I use **vi**) and save the file to $HOME/.Xclients. This won't affect your current X session, but when you log out and log back in the next time, you'll get the new configuration. Quick tip: if you get lost, rename .Xclients to .Xclients.orig and exit your X session. The next time you start your X session, you should be back to where you were previously.

Now you have a simple Xclients file and you're ready to start your X session. If you use a graphical login, you're all set. Your new configuration will start up on your next login. If you use a text console login and manually start your X session, just type **startx**. In either case, should you get lost or confused and want a quick exit, you can type **CTRL-ALT-BACKSPACE** to kill the X server and return to either the graphical login or text console. Quick Tip 2: remember that **CTRL-ALT-F2** should take you to a text login console, and **CTRL-ALT-F7** should take you back to your X session or graphical login.

The first time you start E, you'll see something like Figure 1. E comes with a built-in help system. The large window in the middle of the screen is the Document Viewer, which is used to navigate the help system. This window opens the first time you use E and will open each subsequent time until you exit the Document Viewer. The on-line help isn't difficult to follow, and you should read through it thoroughly at least once. If you look closely, you'll also see one of the informational dialogs from E. This one states that a default menu configuration has been built for you.

Figure 1. Default Desktop

“Enlightenment Basics” described the fnlib library. That library gives E access to some interesting fonts. The Document Viewer, showing the on-line help, shows the default font for this application. This is your first clue as to how E differs from other window managers. You can create some very unusual scalable fonts for use in E. The best way to learn how to do so is to examine the documentation provided with the fnlib source-code distribution.

What happens to the applications started with .Xclients? Most will exit once the .Xclients script exits, and that happens when you exit your window manager. But if some applications don't exit, you can add code (just like any shell script) to .Xclients to force those applications to exit. Remember that in an emergency, you can kill off your X session with **CTRL-ALT-BACKSPACE**, although once you become familiar with E's menu system, you should use the menu options provided for exiting.

Now, back to your first login. What you're looking at requires a little explanation. First, most modern window managers provide multiple virtual desktops. You start out with two desktops with E's default configuration. You can tell that there are two because you have two pagers (lower-left corner of Figure 1). Each desktop can have multiple pages. E's default provides two pages per desktop. The number of desktops and pages per desktop are all configurable using graphical interfaces (i.e., you don't have to edit configuration files manually to change these). We'll talk a little more about configuration a little later, after we've finished our tour of the E environment.

Across the top of Figure 1 is the drag bar. This bar is to desktops what the Title Bar is to windows; it allows you to drag desktops up and down. Unlike other window managers, you can actually view and work in two desktops at the same time with E.

In the lower-right corner of Figure 1 is the default Icon box. It has a slider along its bottom edge, which can be used to scroll icons left and right if there are more icons than can be displayed in the box. Like most things in E, the size of the icons, their arrangement (rows and columns), the number of icon boxes and many other items can be configured using graphical configuration utilities.

A Shady Desktop: Getting More Screen Space

Desktop users often find that their biggest problem is having enough screen space for all the windows they need open at the same time. Linux users are fortunate in that X was designed to make adding virtual space possible in all sorts of ways, and E makes use of nearly all of them.

We've already mentioned pagers, a concept that has been around awhile. Most users have probably heard or seen them even if they have not actually used them. E takes saving screen space a little further by allowing you to “roll up” windows into their Title Bars. The animated effect looks a little like a drawer sliding into a chest. To roll up a window, click the middle mouse button on the Title Bar. Click it again to roll it back out. The animated movement of the window sliding into the Title Bar can be toggled on or off, which is useful on memory-limited systems. When disabled, the window simply disappears, leaving only the Title Bar visible. This rolling in and out of windows is called “shading”.

Shading works for just about any window in E. This includes the pager windows and the icon box. The only windows that can't be shaded are menus. Desktops don't get shaded per se, but they can be slid up and down using the drag bar. The position and direction of slide for drag bars is configurable.

While shading saves you space, you can manage your windows better by grouping them together. Shading a window that's part of a group will shade all other windows in that same group. Grouping windows is done using the Title Bar menu—click the left mouse button on the title to see this menu. Windows belonging to the same Window groups can be iconified, killed, moved and shaded all at the same time. Grouping of windows is a feature found in current versions of WindowMaker and Afterstep, two other popular window managers.

Touring the Desktop: User Menus

We can now take a look at the default menu configuration. I'm assuming you either use a three-button mouse or have your X server configured to simulate three buttons. The latter isn't difficult to do, but we're talking about E here, not the X server. Refer to your X server documentation for how to simulate using a three-button mouse.

Clicking the left mouse button over the root window (i.e., the background) will bring up the Enlightenment menu, which is where you can get access to the rest of the root menus. Become familiar with the Help system option in this menu. You'll be referring back to it on a regular basis.

Figure 2. The Default Enlightenment Menu

Figure 3. The Settings Menu

Clicking the right mouse button over the root window will open the Settings menu. This is where you want to be in order to configure some aspects of E. Options include window focus and auto-raise, desktop backgrounds, and special effects like animated shading of windows and window sliding. The last option causes windows to slide in from one edge of the screen when you change to the page and desktop where they live. Animation options should be turned off on low-end systems and ones with limited memory, to improve overall performance.

Title bars have menus which are opened with right mouse clicks in the title area. This menu allows you to force windows to be “sticky”, which means they are visible no matter what page or desktop is in focus. Here, too, you can set a given window's Window Group, stacking order and border style. You can also use this menu to close windows of applications which may be misbehaving.

Menus are available for many other E features. The icon box's menu can be opened with the right mouse button clicked along its edges or inside the box. One option in this menu is the Iconbox Settings window. Here you'll find many options, including changing the box's orientation (horizontal or vertical), automatic resizing of the icon box, showing or hiding the icon names and backgrounds, and whether or not to use a transparent background in the box.

The pagers have two menus which are also opened with right mouse clicks. Since pagers are just like any other application, their Title Bars (which are just

as likely to be along a side as along the top of the pager window) have a menu like any other application's window. This differs from the pager's menu, opened with a right mouse click inside the pager itself. This menu offers you the option of setting high or low-quality display of windows and images in the pager, as well as a Pager Settings window. This window allows you to set a few pager-specific options. If you're on a low-end system, or one with limited memory (like a laptop), you will want to disable the continuous screen scan option in this window.

Finally, the drag bar has a small menu which allows you to jump directly to windows in other desktops. A middle mouse-button click on the drag bar will show titles of all windows in all desktops. Selecting a title will jump you to that window. If the window is in the current desktop page but hidden by other windows, it will raise that window to the top of all windows. If you left-click on the drag bar and drag down, you can expose lower-layer desktops. A window can be dragged between desktops (onto the same relative page) in this manner. For example, if the window is in the currently visible desktop and you pull the drag bar down a little to expose the next higher level desktop, you can drag the window's title bar into the next higher level desktop. When you release the mouse button, the window's desktop is moved. Again, E provides many different methods for accomplishing the same task. Moving windows between desktops can also be done using various menu options.

The drag bar is just another way of doing what the pagers do, but without taking up any extra screen space. You can disable the pagers completely and just use the drag bar to move around your desktops. The drag bar is long and thin—E works hard to save screen space—and can be configured so it sits along any edge of the display.

By default, E configures menus to be opened using an animated display. On slower systems or systems with limited memory, it is better to disable this feature using the Settings menu (right-click in root window).

Touring the Desktop: Pagers

Virtual screen space gives you added room to keep windows open. In essence, they are an alternative to iconizing windows. Icons are nice, but without meaningful icons to represent the real window's application, icons can be more confusing than helpful.

Pagers get around this ambiguity by simply leaving the windows open. You jump to and fro between virtual desktops with multiple pages. Think of each desktop as a desk in a different cube with multiple drawers. Each drawer would be a page in the desktop. The main difference is that it's easier to work on the virtual desktops than it is to swap papers to and from the desk and drawers.

E's pagers are fairly sophisticated, compared to pagers for other window managers. First, the background image used on a desktop can be displayed in each page of the pager, making it easier to delineate the pages. Each window in a pager is a miniature version of the real one, complete with window borders, window contents and images. The miniaturized windows in the pager can be zoomed when the mouse is over them, although for things like ordinary xterms, this isn't very helpful. What is useful is having the window title displayed when the mouse is over a window in the pager. This allows you to identify any window in any page of any desktop instantly—and a simple middle mouse button click on that window in its pager will jump you right to the window.

Having different backgrounds for each desktop is also supported. Figure 4 shows a desktop configured with 3 pagers along the right side, each with a different desktop background image and a 2x2 page configuration. A Netscape image and multicolored sticky notes are visible in various pages.

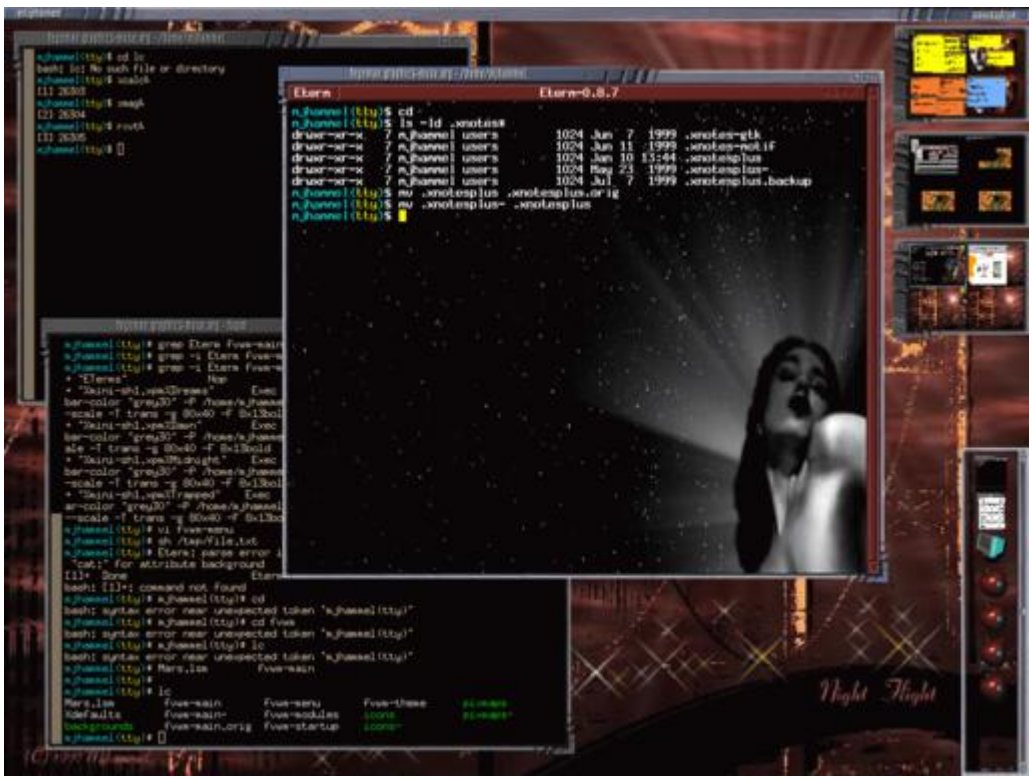


Figure 4. Example of Multiple Pagers

This example shows a modified configuration using a theme different from the default, personalized backgrounds and pagers with different configurations. Compare Figure 1 with Figure 4, and you'll see just how varied the E environment can look.

Epplets

One of the newer features of E is its ability to support small programs, called epplets, to perform simple functions. The epplets are usually small C programs,

but you can also create small scripts run by eesh, the Enlightenment shell. Epplets can be stand-alone programs which have their own user interface, or they can let E handle the user interface for them—a sort of embedded application.

Currently, there are only a few small epplets, provided mostly by loyal E enthusiasts. E itself comes with a few scripts, but no epplets. None of the epplets do anything overly interesting, although it's highly likely that functions like automatic connection to the Internet will be provided in the future using epplets. The most interesting of the epplets is probably **gkrellm**, an all-in-one epplet that displays system performance, time and other useful information. You can expect that future extensibility of E will come in the form of expanded support for epplets.

Personalizing the Desktop

After all this, why should you consider E? In a nutshell-- configurability. E offers you the most options to making the desktop yours—your style, your feel, your world. You live on computers day in and day out. Why shouldn't it be a more personal place?

E has a considerable amount of graphical configuration support. In fact, nearly all features of E are configurable using simple graphical interfaces that you can access via the Settings menu or from feature-specific menus. The only parts of E that require manual intervention are adding menu options to the root menus and developing themes.

The Settings menu allows you to change the way the window manager works in many ways: opaque vs. transparent window moves, autofocus and autoraise for windows, desktop, tooltip and audio settings, and even a few special effects. It's important to scan through these settings, especially if you're running on a memory-limited laptop. Enlightenment is designed to allow you to add more flash to your desktop if your system has the CPU and memory to deal with it. But if it can't, E also allows you to disable some of the resource hogs.

Changing the root menus is fairly easy. Just edit the file.menu and/or user_apps.menu files under your $HOME/.enlightenment directory, which was created the first time you started E. You can add new submenus for the root menus in this way. The file.menu file corresponds to the User Menus menu (opened with a left mouse click on the root window). The user_apps.menu displays as the User Applications List menu, an option from the “User Menus” menu.

E checks the entries in these menu files to see if the program associated with a menu option can be found. If the program can't be found, then that entry won't

be displayed in the menu. You can use fully qualified path names for programs. If the application is in a directory listed in your PATH environment variable, you just need to specify the program name.

Changing menus lets you have quick access to starting applications. How do you force an application to always start on a particular desktop page? E can remember open applications, but you have to specify manually which windows to reopen the next time you start an X session. To do this, hold down the **ALT** key while you right-click in a window; a menu will open. Select the "Remember..." option, and in the dialog that opens, select the options you want E to remember for this application. The next time you start E, it will start the application with the parameters you specified.

Note that this differs from how we started X the first time, using the .Xclients file. That file is used by xinit to start applications every time you start an X session—Enlightenment doesn't know anything about those applications. The **ALT**-right mouse click method just mentioned tells Enlightenment to restart an application after E is started. The choice here is whether to allow xinit or E to start your applications. To make life easier, the only line you want in your .Xclients file is the command to start Enlightenment; then, use E to manage the starting of any other applications.

Dragging windows can have many effects. Dragging a window from a pager over the root window will drop that window onto the current page and desktop. You can drag windows from the pager into the icon box to iconize the window without actually going to the desktop the window lives in. The icon box can be resized by holding down the **ALT** key and clicking the middle mouse button—not intuitive, but easy enough once you know what to do.

The background for any desktop can be set using the "Desktop Background Settings..." option in the Settings menu. User's backgrounds in just about any format (for example, TIFF, JPEG and PNG as long as your version of imlib was built to support these) can be copied to ~/.enlightenment/backgrounds. Changes to this directory will be recognized only if you restart E; however, this is easy to do using the "User Menus" menu.

How would someone go about figuring out how to change the themes? How do you find out how to create stylized buttons, Title Bars and so forth? This is the current limitation in E (and, actually, in many other Linux desktops). Simple themes can specify background images for windows and desktops, and images to use for icons. E allows more sophisticated control of the display by allowing you to specify window borders, title bars, edges, corners and various other bits and pieces. The bad news is there isn't yet any documentation anywhere that accurately describes how to do this. The good news is that whenever

something minimal is provided by the developers, I'll be able to turn that into something more detailed for you.

Despite the lack of information on creating your own themes, many prepackaged themes are available for E from the official site for Linux themes: Themes.org. This site has sections devoted to the various window managers which support some form of themed environments. I looked through a number of the user-contributed themes for E that you can find here, but didn't find any that were visually more appealing than the BlueSteel theme provided in the default installation of E. Beauty, of course, is in the eye of the user.

Performance Issues

Most of my research for this article and others in this series has been done on two systems: a Pentium II 400MHz box with 256MB of memory and an IBM ThinkPad i1410 (Celeron 200MHz) with 32MB of memory. By using two significantly different systems, I was able to compare some visible performance differences for the various window managers.

Performance on the Thinkpad was reasonable, even with animations turned on. The slide-show change to different desktops is a little jumpy, but not really slow. Translucent moves—windows appear partially transparent when being dragged around the desktop—were extremely slow, and shaded moves were nearly unusable. Use Shaded moves only if you plan on shading the windows first or have a fast CPU. Technical moves are fairly interesting visually and quite fast, since they draw only the boundaries of the windows. Box moves aren't as visually stimulating, but are quite fast. Box and Technical moves are the options of choice for slower, memory-limited systems.

Another thing to look at for performance issues is the Imlib Configuration tool. By default, you can get to this from the User Menus-->User Application List--> Imlib Settings option (left mouse click in the root window). From here, you can set the color quality of the window rendering, and the amount of cache to use for images and pixmaps. The Color Correction page is quite interesting, although its effect is not obvious unless you have high-quality background images. On lower-end systems, try reducing the Image Cache, Pixmap Cache and Shared Memory sizes to their lowest settings. Also, disable the High Quality mode for 15/16bpp systems. Many laptops are likely to run in 15/16bpp, so you'll want the High Quality mode disabled on these systems.

In the End...

E is still in development, so it may possibly cause system lockups. I tried to access the Audio features from the Settings Menu, even though I had turned off the sound options during compilation, and this caused a segmentation fault.

Interestingly enough, E caught this problem and offered me the option of restarting or continuing. Although the fault is a problem, it handled it gracefully.

E is cool and more than likely has a very interesting future to come. It's fairly stable—I had only two lockups in all my testing on both systems—but lacks end-user information to make serious use of all its features. It comes with one of the most interesting default interfaces available. Many people are recognizing the potential of E. It works happily in both the GNOME and KDE environments.

E is also resource-intensive—it can chew up memory fairly quickly. Because it's still in development, installation can be a pain for the inexperienced user. You need to understand how to build and install multiple software packages just to get it running properly. Configuration of menus can be complex and requires manual editing. Stylized, themed interfaces are possible, but it's not yet clear how to create these. There isn't any real documentation that begins to explain the process.

My purpose here was to talk about what the desktop environment can be to end users—distinctively personalized. E offers the potential for this more than any other window manager, but it's a long way from the simplicity that typical desktop users expect. Its complexity notwithstanding, if you've got the determination to experiment and research the default configurations on your own, you may find that E will have its place in your world.

X Session Technical Description

Michael J. Hammel (mjhammel@graphics-muse.org) is a graphic artist wanna-be, a writer and a software developer. He wanders the planet aimlessly in search of adventure, quiet beaches and an escape from the computers that dominate his life.

Archive Index Issue Table of Contents

Advanced search

Transmeta Rewrites the Rules

Linley Gwennap

Issue #72, April 2000

All about the revolutionary new chip from Transmeta.

When Linus Torvalds joined the secrecy-shrouded startup Transmeta Corporation, he couldn't tell anyone what he was working on. Now, the wraps are off a revolutionary new idea: a chip that combines the low power of a RISC chip with the software versatility of an x86 processor. This combination enables a new class of lighter, longer-lasting mobile devices with PC compatibility, and yes, some of these devices run Linux. The new chip (see Figure 1), dubbed Crusoe after the famous traveler, achieves this combination of features without any RISC or x86 hardware. Instead, it runs so-called code-morphing software on VLIW (very long instruction word) hardware to execute standard PC software. This unique combination delivers the low cost, low power and software compatibility needed in a mobile PC or a portable web pad.

Figure 1. Crusoe Chip

VLIW Technology Reduces Cost, Power

Figure 2. Dave Ditzel

Dave Ditzel (see Figure 2), a former Bell Labs and Sun CPU architect, founded Transmeta in 1995 to pursue a novel approach to x86 microprocessor design. Whereas Intel and AMD take power-hungry desktop PC processors and cripple them to fit into a notebook PC, Transmeta decided to design a new mobile x86 processor from the ground up. As a result, the company's Crusoe chips dissipate as little as one-fifth the power of Intel's Mobile Pentium III. To minimize power, Crusoe uses a simple yet powerful VLIW architecture that requires far fewer transistors than Pentium III or AMD's Athlon. These traditional chips execute several instructions at once to improve performance. Standard x86 programs, however, assume the processor executes one instruction at a time. Therefore, a huge portion of a Pentium III or Athlon chip is devoted to checking and arranging instructions so they can be safely executed together.

A VLIW design, such as Crusoe, avoids this overhead. Each Crusoe instruction contains the equivalent of four x86 instructions, all prearranged for optimum execution. With no checking or reordering logic, the processor can execute all four instructions at once and immediately begin the next set of four instructions. As a result, the processing logic in Crusoe requires 75% fewer transistors than in Pentium III. This difference provides several advantages. Each transistor on a chip consumes electricity; with far fewer transistors, Crusoe needs less power to operate. The missing transistors also make Crusoe

less expensive to build than a Pentium III. The simpler chip requires less time to design and is easier to test—important issues for a small company such as Transmeta.

Code-Morphing Delivers Compatibility

Alone, a VLIW chip is incapable of running programs designed for an x86 processor. But the Crusoe chips are paired with Transmeta's patented code-morphing software (CMS), which converts x86 programs into VLIW instructions that the chips can understand. Although Transmeta is a chip company, developing this software was a more daunting task than developing the chip itself.

The CMS, which Torvalds helped to develop, analyzes an x86 program and feeds the corresponding VLIW instructions to the processor. This translation is done on the fly and is invisible to the user. The CMS translates only those portions of the program that are in use; for example, when executing Microsoft Word, the “mail merge” module is not translated unless it is invoked.

Translated code blocks are saved in a translation cache; on subsequent iterations, no retranslation is needed. Since programs often execute the same loops and subroutines over and over, the translation cache is very effective. But the CMS has the odd property that it executes a program relatively slowly to start, but faster over time. Translations are not saved when the user quits an application, so the learning process starts anew each time the application is launched. The overhead of code morphing reduces the raw performance of the VLIW engine, but over time, this overhead is amortized across many iterations of frequently used code blocks. Transmeta's VP of Product Development, Doug Laird, claims that the 667MHz Crusoe chip will match the performance of a 500MHz Pentium III, although the company has not released benchmark data to support this claim.

Many popular PC benchmarks use a particular feature of an application very briefly before moving on, reducing the advantage of the translation cache. This problem makes measuring the performance of the Crusoe chip difficult. Real users, in contrast, typically use the same features again and again. For Transmeta to compete well against Intel, end users must perceive Crusoe to be just as fast as Pentium III.

Hardware and Software Designed Together

Processor designers and software designers typically work for different companies. If they do work at the same company, they rarely deign to speak to one another. At Transmeta, these engineers work hand in hand. The unique

combination of hardware and code-morphing software allows them to solve problems in ways no other company can.

This is because the CMS is the only code that uses Crusoe's VLIW instructions; all other software for the chip is written in x86 code. Therefore, the processor designers are freed from the burden of maintaining compatibility; in fact, the first two Transmeta chips are not even compatible with each other! Each uses its own version of the CMS, and both are fully compatible with all x86 programs through code morphing. If the hardware team wants to give the chip a feature that improves performance or reduces cost, they need only clear it with their compatriots in CMS land, and the change can be made. Conversely, a bug in the hardware design can often be solved by telling the CMS “don't do that”, avoiding a costly and time-consuming redesign of the chip.

The biggest fear of an Intel CPU engineer is that their design will have a bug that prevents some application from running correctly. The infamous Pentium floating-point bug cost the company hundreds of millions of dollars in 1995. In contrast, if the CMS doesn't execute an x86 application correctly, the CMS can usually be patched and reissued within a matter of days. Crusoe may be the first microprocessor to be upgraded in the field. CMS engineers also ask for and receive hardware changes that accelerate x86 execution. Previous attempts to implement x86 compatibility in software, such as Sun's WABI (x86 on SPARC) and Digital's FX!32 (x86 on Alpha), failed because they didn't offer the right combination of 100% compatibility and competitive performance. Neither SPARC nor Alpha offers any support for x86 in hardware, whereas the Crusoe chips include small but important circuits that speed the code-morphing software beyond what previous efforts achieved.

Competing in Mobile PCs

With its low power and x86 compatibility, Crusoe is an obvious fit for a notebook PC. The processor runs Microsoft Windows and standard PC software, just like a system that uses an Intel or AMD processor. Its low power extends battery life and reduces the amount of heat the processor gives off. With a cooler chip, notebook makers can take out fans and heat sinks that add weight, burn power and make noise. Not satisfied with the inherent benefits of its VLIW design, Transmeta developed its LongRun technology to further trim Crusoe's power consumption. During operation, the CMS monitors the workload of the processor and dynamically adjusts its clock speed to match. Unlike other chips, Crusoe can reduce its voltage as well as its clock speed; because CMOS power equals cv<+>2<+>f, at 50% workload the chip needs less than 20% of its peak power.

LongRun was announced the day after Intel's SpeedStep technology, which also changes clock speed and voltage in mobile systems. SpeedStep processors go

faster when the notebook PC is plugged into the wall, but SpeedStep has no effect in battery mode. In contrast, LongRun conserves power, extending battery life when a notebook user is on the go. The low power of Crusoe will extend the battery life of traditional notebook PCs, but only by 30% or so in most situations. That's because the CPU typically consumes about a quarter of the battery's output, with most of the rest used by the display backlight, hard drive and CD-ROM drive. Transmeta likes to focus on applications, such as DVD, that are more CPU-intensive and therefore have a bigger advantage on Crusoe. But simply adding Crusoe to today's notebook PCs will not result in an eight-hour battery life.

Creating a New Class of Devices

Crusoe's power advantage is more critical in the emerging market for portable web devices. In a wireless web pad with no drives (see Figure 3), the CPU is a large part of the power budget. Intel suggests its Celeron processor for these devices, but Celeron burns too much power for extended battery life. Intel's StrongArm chip uses even less power than Crusoe, but doesn't offer x86 compatibility.

Figure 3. Prototype Web Pad Using Crusoe Chip

Why is x86 compatibility important? After all, these devices don't run Windows or Windows software; they use an embedded operating system, in some cases the new Mobile Linux, and a browser that has been compiled specifically for the internal processor.

Transmeta points out that many browser plug-ins are written in x86 code. To get the full web experience, a web pad must be able to download and execute new plug-ins. Without these plug-ins, some web pages simply won't work. To offer the same experience for a StrongArm web pad, a system vendor would have to fund the porting of each new plug-in to the StrongArm instruction set.

Instead, Crusoe offers out-of-the-box compatibility with all web content. In addition, it gives the system maker the choice of Linux, Windows CE, or any other operating system that already runs on the x86 platform. Crusoe also supports all of the popular x86 software-development tools. The chip's low power and x86 compatibility give it a clear advantage in this market segment.

The bad news is that the market for web pads is basically nonexistent, limiting Transmeta's potential sales. Crusoe burns too much power for today's most popular portable web device, the Palm VII. But this market could grow to tens of millions of units over the next five years, and Transmeta is poised to grow along with it.

By boldly combining hardware and software in a new way, Transmeta has built a processor with clear advantages in power and compatibility. If Transmeta delivers on its performance claims, Crusoe should launch a new class of devices which let you take the Web wherever you go. With their code-morphing software, Crusoe devices will run nearly any operating system. But as Transmeta's Torvalds says with a smile, they will run Linux just a little bit better.

email: linley@linleygroup.com

Linley Gwennap (linley@linleygroup.com) is the founder and principal analyst of The Linley Group (http://www.linleygroup.com/), a technology analysis firm in Mt. View, California. He is a former editor in chief of Microprocessor Report.

Archive Index Issue Table of Contents

Advanced search

NetMAX Apache WebServer

Allan Liska

Issue #72, April 2000

Unlike most Net appliance makers, NetMAX concentrates on software, leaving the hardware for the user to supply.

- Manufacturer: Cybernet Systems Corporation
- E-mail: info@netmax.com
- URL: http://www.netmax.com/
- Price: $89 US
- Reviewer: Allan Liska

With the self-stated goal of “simplifying Linux”, NetMAX has introduced a trio of “Net appliance” products: WebServer, FireWall and FileServer. You can also get all three rolled into one, along with additional network management tools, in the Professional version of the software. Unlike most Net appliance makers, NetMAX concentrates on software, leaving the hardware for the user to supply.

NetMAX makes the software available in both Red Hat Linux and FreeBSD versions, both of which, they claim, can be installed in under 15 minutes. As with most appliance software, common administrative functions can be performed through a web browser.

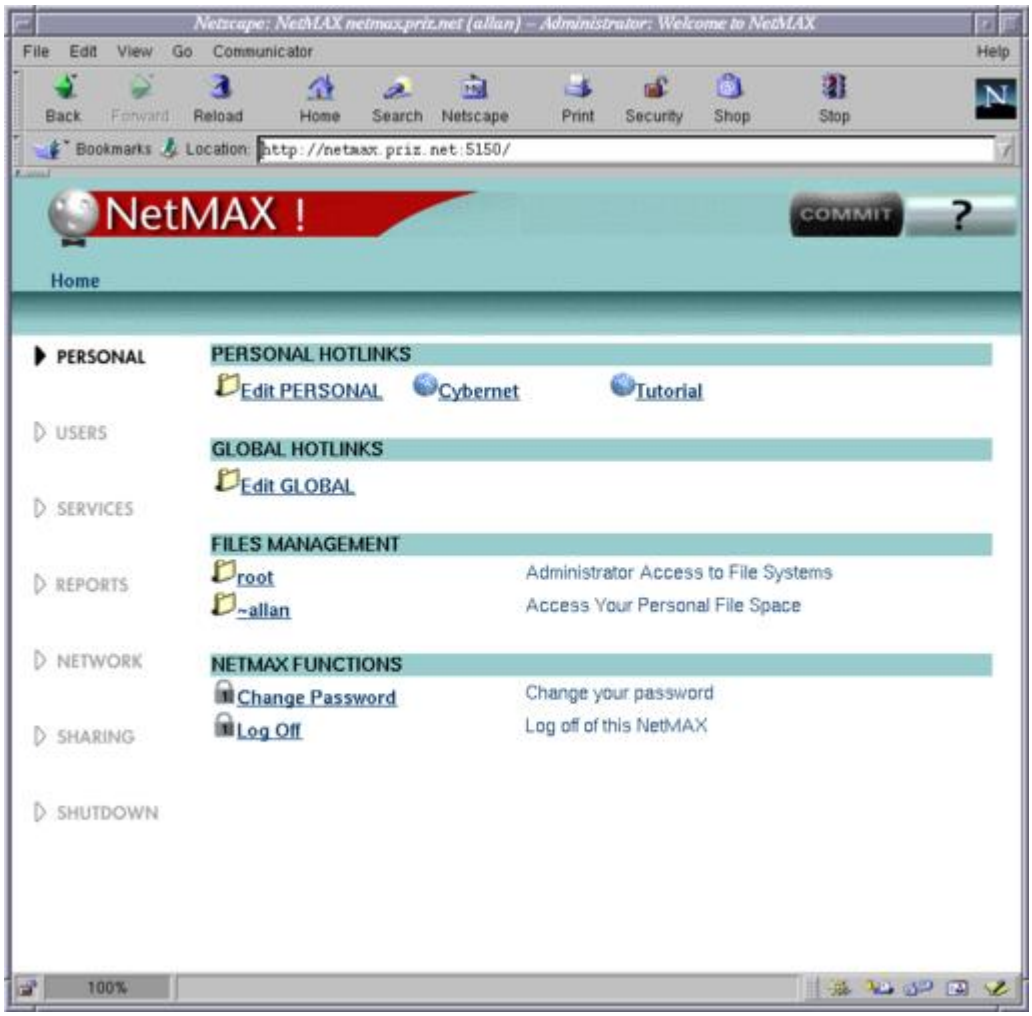


Figure 1.

The NetMAX WebServer can be installed over an existing Red Hat system, or it can be installed fresh. I chose the latter option, ordered the NetMAX WebServer, and built the following machine to support it: AMD 300, 64MB of RAM, 2GB Seagate IDE Drive and Intel 10/100 Pro Ethernet Adapter. All of these are well above the listed minimum requirements of a Pentium-based processor: 32MB of RAM and 1GB hard drive. It takes the local Chinese food place about 25 minutes to deliver an order, so I decided to use them as a timer. I placed my order, opened the box, and began the installation process.

The WebServer comes with a boot disk, a CD with the software and two manuals: Installation and User. Having done dozens of Linux installations, I bypassed the manuals, put the CD into the CD-ROM drive and booted the machine. The installation screen appeared, detected the LAN card, and asked me for an IP address and netmask, while files were being copied over to the hard drive. I entered them and was informed I could finish the install from the terminal or over the Web, by going to the entered IP address on port 5150. This part of the process took about five minutes.

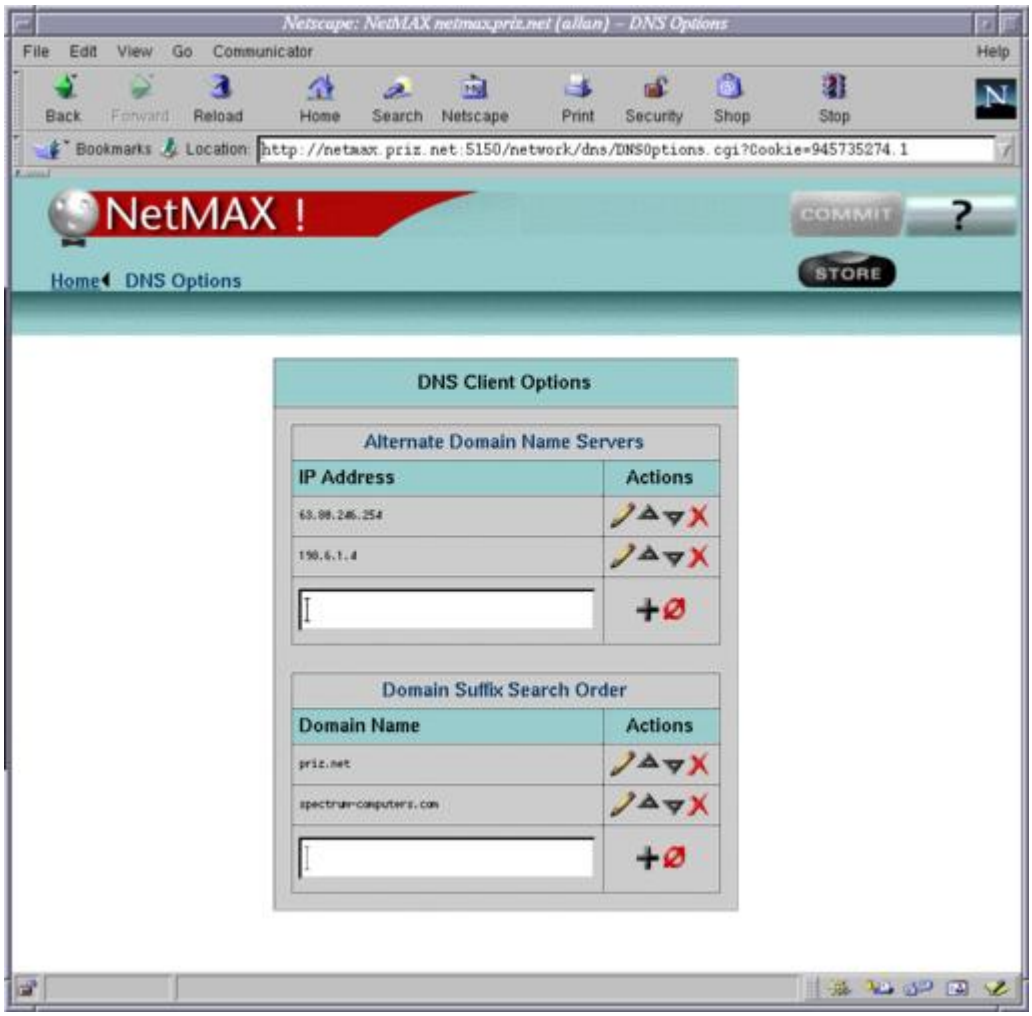


Figure 2.

I opted for the web installation, sat down at a nearby machine and accessed the IP address at port 5150 in a web browser. The admin page came up promptly, I was asked a couple of questions about the network configuration, and the installation began. Total time so far: ten minutes.

The only snag I ran into with this process was the WebServer did not detect our local nameserver for some reason. I substituted another provider's nameserver, it was detected, and the install continued. After the install, I went back and replaced the other provider's name server with ours, and it worked fine.

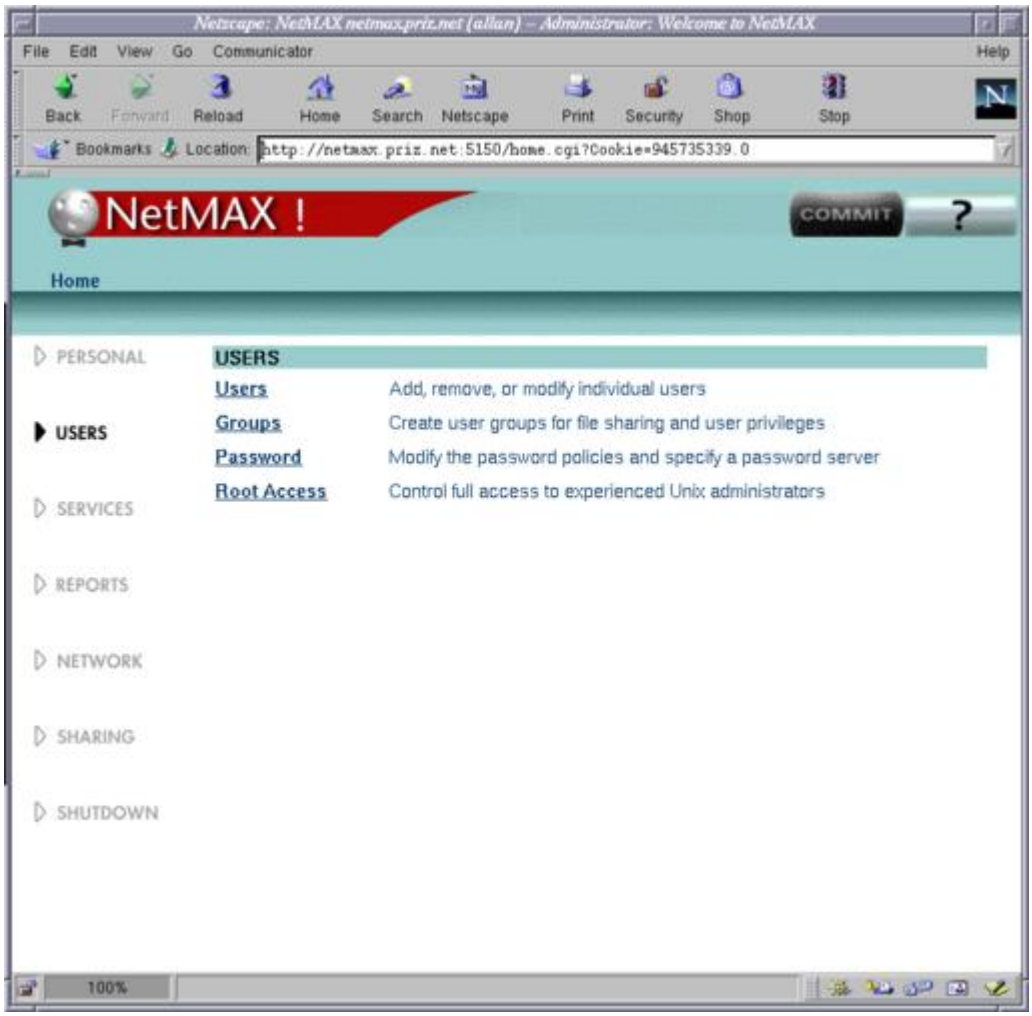


Figure 3.

Everything was done automatically; the WebServer software formatted the drive, created the partitions and installed the necessary services. It even automatically detected the local Windows Workgroup name and added itself to it, sharing the root HTML directories automatically so they could be updated easily by anyone in the organization with proper access. While the software is installing, the browser displays a status bar, tracking the entire process and updating you with what is being done.

The delivery person arrived just as the machine was finishing and getting ready to reboot. Total time: 25 minutes. While this was not quite the advertised 15 minutes, it certainly was not too long to wait to have a fully functioning web, mail and FTP server.

As I ate my Kung Pao chicken, I explored the web interface to the WebServer. I was a little disappointed to notice that the version of Red Hat installed by the NetMAX WebServer was 5.2. A quick check to the NetMAX web page told me the new WebServer version would be shipping in mid-January, and as a registered user, I would be entitled to a free upgrade. Another disappointment I had was

that the server did not come pre-installed with PHP or MySQL. Fortunately, these were very easily installed.

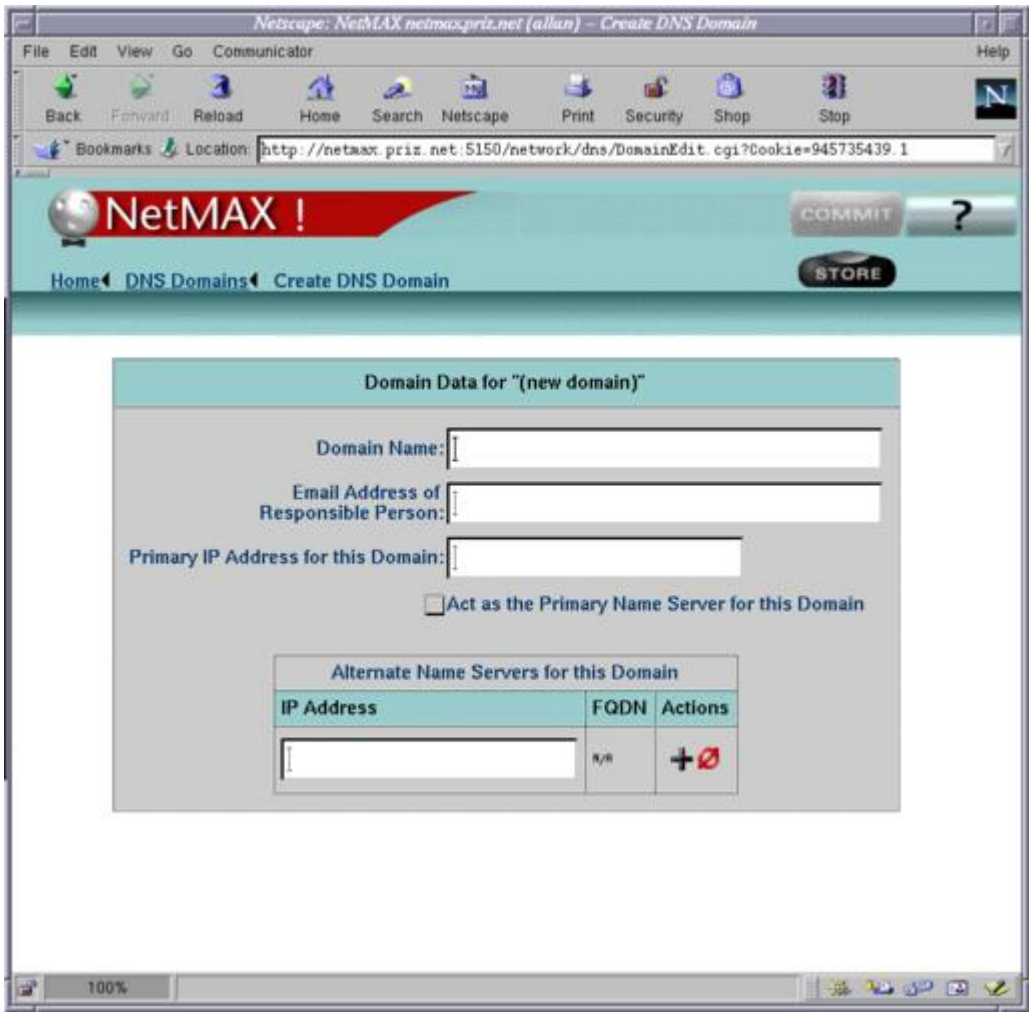


Figure 4.

The WebServer software packages include Apache, with optional support for FrontPage Extensions, Sendmail and WU-FTPD. There is also an X Window System Client and a remote X-application server.

The web interface is very impressive. It is a series of CGI scripts that allow you to update and change a wide array of daemons on the server. Our company has been using the Cobalt Network's RAQ Servers for more than a year, so I was prepared to find similar features with the NetMAX WebServer. Instead, I found a much wider breadth of services.

After you log in to the WebServer, you are taken to a framed Welcome page. From this page, you can take a tutorial of the interface, or you can begin to make changes. On the left side of the page is a menu with seven choices: Personal, Users, Services, Reports, Network, Sharing and Shutdown.

Most experienced Linux administrators should be readily able to determine what is available under each menu, but even Linux novices should have no problem finding what they want.

Obviously, there is not space to go over the content of each sub-menu, but some of the significant things you can do from the web interface include set up e-mail accounts, turn the server into a router, enable and disable remote FTP access (by default, it is disabled from outside the LAN), add new domains and set up primary and secondary DNS for those domains, access CD-ROM and floppies, install new packages (RPMs) and reboot the server.

For all the many useful features the server has, the extensive coding does have a tendency to slow it down somewhat—but only when working within the administrative area; the actual web server responds very quickly. Experienced users may find it quicker to make the changes from the command line; NetMAX anticipated this, so any changes made from the command line override changes made through the GUI. However, inexperienced users will definitely find that the administrative control gained is worth any latency they may experience.

One other nice security feature is that the WebServer will automatically log you off your administration session if it has been idle for too long.

If you would like to get a feel for the administrative server and what features it provides, NetMAX does have an excellent demo on their web site. It's worth taking a look at.

Allan Liska built his first computer when he was 12—a lovely Heathkit with a whopping 16K of memory. He started Spectrum Computers with his wife in 1998, which has since evolved into Priz*Net (http://www.priz.net/) Internet Services—all Linux-based. The important part: his first experience with Linux was in 1995 with Red Hat 5.1. Since then, he has been an avid Red Hat fan. He plans to keep his Windows box around until someone ports Duke Nukem and Test Drive to Linux. He currently runs an AMD 450 with 64 megs of RAM and Red Hat 6.1.

Archive Index Issue Table of Contents

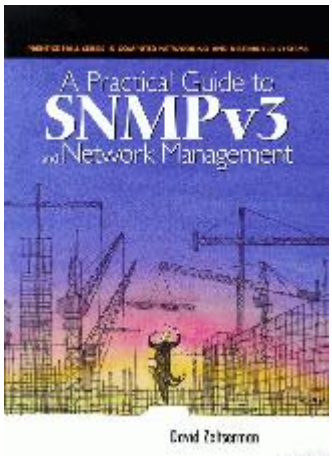
Advanced search

A Practical Guide to SNMPv3 and Network Management

Charles Curley

Issue #72, April 2000

The book assumes a cursory familiarity with SNMP, but does not require any knowledge of the underlying protocols (UDP, IPX, etc.).



- Author: David Zeltserman
- Publisher: Prentice Hall
- URL: http://www.phptr.com/
- Price: $54.00 US
- ISBN: 0-13-021453-1
- Reviewer: Charles Curley

Prentice Hall is very good at issuing books which are very good within a narrow specialty. You can use the book intensively and learn much from it, in which case it is worth what you pay for it. If you are only casually interested in the subject, the book isn't worth the price of admission and you should look for something else. *A Practical Guide to SNMPv3 and Network Management* is one of those books.

Engineering students working on SNMP (simple network management protocol) and engineers designing and implementing SNMP entities (devices and network

management software) will benefit the most from this book. If you are a network administrator who uses SNMP engines but you don't "get under the hood" very often, then this book is probably overkill. It is aimed specifically at SNMP version 3, with historical references to older versions of SNMP. If you need a more general introduction to SNMP, the classic one is *The Simple Book* by Marshall T. Rose, now in its second edition.

The book assumes a cursory familiarity with SNMP, but does not require any knowledge of the underlying protocols (UDP, IPX, etc.). There is an introduction to SNMP, so if you have never dealt with versions of SNMP prior to 3, it will bring you up to speed with a historical perspective. Since any network of reasonable size will have a mixed bag of SNMP versions on it, this historical perspective is essential.

The book builds on some basic blocks, such as a detailed description of the character set allowed for the data type **DisplayString**. There is a table of control characters indicating what they do. These basic building blocks may be tedious to read about, but they are essential if you want to avoid misunderstanding.

Much of the text is accompanied by pseudo-code, written in the author's own pseudo-code language. The language is a bit disconcerting to start with, but easy enough to learn. The pseudo-code examples are detailed, well-commented and occasionally extend across multiple pages. The pseudo-code examples should port readily to C.

The author divides SNMP into three areas. First is the SNMP protocol itself. This consists of the messages SNMP uses, their formats and how SNMP entities interchange them. Second is the Structure of Management Information (SMI), a set of rules for ordering information on a managed device. Third is the structured collection of information on a managed device or a management information base (MIB).

After the introduction and a chapter on the basics, the author takes us on a tour of MIB-II, as modified by experience, if not formal standards. We are presented with an item-by-item list of the entries in MIB-II. However, the author also gives his own opinions from time to time. I'm glad to have the benefit of Zeltserman's 16 years of experience building networks and network devices.

Sometimes, however, the author's experience gets in the way. MIB-II is divided into several groups, one of which is the EGP group. He simply informs us that it isn't used any more, so he doesn't describe it. Given the thorough detail in most of the book, that terse statement is disconcerting. It's okay for folks implementing a brand-new SNMP entity. But for those readers who might have to deal with an older entity, it would be nice to know why the EGP group is no

longer used. Even better would be to document it and say why it isn't used any more, leaving the decision to the reader. Any serious reader of this book has access to other literature and especially to the RFCs, so this is certainly a deficiency the reader could quickly remedy, if need be.

The next chapter introduces the architecture of the SMNPv3 framework. It introduces some concepts new to SNMPv3, particularly the modularity of the framework, and new security and administration features.

One portion of SNMPv3 engines consists of applications for such things as generating and responding to commands and notifications (formerly traps). This is the subject of Chapter 5.

Chapter 6 begins coverage of security, which SNMPv1 veterans will find changed and considerably enlarged. Authentication and the data encryption algorithm are discussed.

Chapter 7 covers view-based access control. This is a security technique that allows the administrator to determine which users may see and modify certain views into an MIB.

Chapter 8 deals with coexistence. This is the problem of translating from older versions of SNMP to SNMPv3. This becomes problematic either when a proxy may have to translate, or when an entity supports multiple versions of SNMP.

RMON2 is covered in Chapter 9. Where previous versions of RMON dealt only with the physical address (MAC address for Ethernet), RMON2 goes to higher-level protocols and lets you analyze traffic for specific protocols.

A shortcoming common to many technical books is the failure to explore the implications of the subject for personal privacy, a right rapidly vanishing from the Internet. Because RMON2 can look at an individual computer's application layer network traffic, it has clear implications for personal privacy at home and work. This, in turn, brings up ethical questions, such as: do network administrators want to provide proof that Joe is looking at porn web sites—or Dilbert—on company time? Is it our business that Sue is using e-mail to set up an assignation with a man not her husband? Yet, the author provides no warning to that effect.

The final chapter describes a number of Cisco private MIBs. Cisco has done an excellent job of documenting their private MIBs, and has a large share of the market, so the Cisco private MIBs are an excellent choice. Still, a comparable HP MIB or two, say, would have made an interesting comparison. The gist of the

chapter suggests how the network manager can take advantage of private MIBs.

One private MIB the author explores is the Ping MIB, which like the TCP/IP networking utility of the same name, lets you measure instantaneous network response times. This has clear implications for network tuning. This and other private MIBs may lead one to speculate that beneath the dull exteriors of some Cisco boxes beats the heart of a penguin.

The book has an index. It appears to list every object described in the book, which makes it an excellent reference work. However, some concepts are missing, and that makes the book harder to use as a high-level reference work.

The bottom line: if you are serious about SNMPv3, get this book.

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

Archive Index Issue Table of Contents

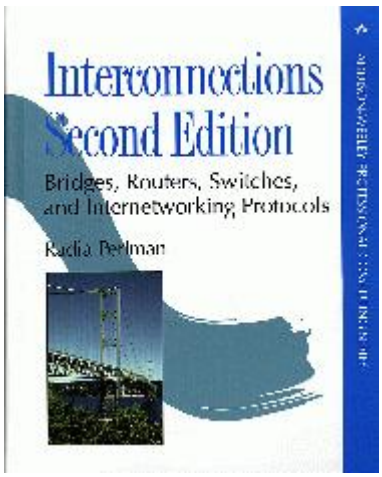
Advanced search

Interconnections, 2nd ed.

He Zhu

Issue #72, April 2000

One of the best books available on networking today.



- Author: Radia Perlman
- Publisher: Addison-Wesley
- E-mail: cepubprof@awl.com
- URL: http://www.awl.com/
- Price: $59.95 US
- ISBN: 0-201-63448-1
- Reviewer: He Zhu

Interconnections was written by Radia Perlman, a leading networking expert. She wrote this book intending to help people understand networking problems and provide a reference to relevant issues. She didn't simply present a collection of texts condensed from the literature and specs; instead, by keeping an open mind, she documented many techniques, both good and bad, in a carefully arranged manner with an expert's insight. It is one of the best books available on networking today.

People surf the Internet, from which they find an explosion of sources of human wisdom and civilization. Behind the scenes of the Internet are specialized boxes and protocols. These boxes are called bridges, routers, switches and hubs, and are connected via various communication channels, through which they talk with each other in predefined languages known as protocols. Writing a book to describe this area comprehensively is a daunting task. We haven't yet seen many good books in the field. *Interconnections* is excellent in its coverage of issues in both layer 2, that is, the data-link layer on which bridges are operating; and layer 3, the network layer on which routers are operating. It presents a broad view of the techniques to connect computers together. The book gives an excellent introduction to all these issues.

The first edition of this book, published in 1992, has been a best-seller and acclaimed as a classic on the topic. Since then, tremendous progress has been made in internetworking technology. Traditional companies are feeling pressure from new start-ups in the networking arena. More people are eager to learn about networking. It is great to see a new edition of *Interconnections* published at this time. The second edition is not simply an update, but a competent rewrite of most chapters, with a few exceptional chapters remaining largely unchanged. This new edition places more emphasis on networking designs and covers issues not found in the first edition, such as hubs, fast packet forwarding, autoconfiguration, multicasting and protocol design folklore.

This book is comprehensive for general networking concepts and techniques, including obsolete techniques in legacy systems such as X.25 and up-to-date advances such as multicast routing. The author intentionally puts various networking techniques (both good and bad) in one place to help people understand not only why we do something in this way but also why not in that way, in hopes that future networking designers can learn from these lessons and avoid the same mistakes. Given a problem, the author doesn't simply list a well-accepted solution, but often provides several alternatives. This helps reveal the differences, advantages and weaknesses of various solutions. For example, in Chapter 9 the author summarizes almost all well-known network layer address structures, whether widely or seldom used: IP, IPX, IPv6, CLNP, AppleTalk and DECnet.

Interconnections are implemented mainly by bridging and routing. The book discusses issues around these two interconnection approaches. It also mentions connections by devices like hubs, which are simple repeaters and widely used in offices. The best part of this book is the section on bridging. One of the author's greatest contributions, the spanning tree algorithm, was made in this area. Routing is complex. The author also did an excellent job in summarizing this issue. Most chapters are devoted to routing. Compared with

other books on routing techniques, this one covers more general topics and has more author's comments. Moreover, Perlman's book is outstanding in that it deals with both bridging and routing in one text. Bridging techniques rarely found anywhere else are described in depth.

Confusion in networking terminology often arises in the networking industry. This is due to the complexity of the problems and, historically, the lack of consensus. The author makes complex topics understandable by using simple words and examples. The author tells us the word “switch” means nothing but “fast” bridge or router or some hybrid, better used as a generic term for a box that moves data. She also relates anecdotes, which make the text more entertaining as well as more informative.

Like many other books on networking, this one starts with an introduction to the well-known, seven-layered OSI reference model. The first chapter is characterized by discussions of a few networking properties which are important in network designs.

I found chapters 2, 3 and 4 to be the most interesting part of the book, as with the first edition. They are excellent descriptions of bridging techniques. The author describes transparent bridging and her invention of the spanning tree algorithm in detail. As an alternative (although not popular) solution, source routing bridges are introduced in Chapter 4.

Chapter 5 briefly discusses devices evolved from original repeaters, bridges and routers like hubs, faster and virtual LAN techniques.

Chapters 6 to 15 are about network layer (that is, layer 3) techniques. These chapters cover introductions to almost all major issues about moving data around internets, or simply routing. In these chapters, readers can find a wide range of information on networking techniques such as network service models (like connectionless and connection-oriented, best-effort and reliable services), address structures (like IPv4 and IPv6) and routing protocol data packet formats, and unicast and multicast routing protocols (like RIP, BGP, OSPF, ISIS, PNNI, DVMPR, MOSPF and PIM-SM).

Chapter 16 is an abstraction of the author's PhD dissertation on a technique for a high degree of network robustness.

Chapter 17 and 18 summarize the book. Chapter 17 compares bridges and routers. It further clears the confusion between terms of bridges, routers and switches. Chapter 18 is completely new in the 2nd edition. The author illustrates some good protocol design philosophy by relating some folklore. I thought it was the most entertaining chapter of the book.

The Internet infrastructure, consisting of interconnected devices and information sources, is a must for Linux to flourish. Linux, in turn, has been making ever-growing contributions to interconnection technology. Although not a single word about Linux can be found inside, *Interconnections* should benefit anyone inside and outside the Linux community, beginner and expert, developer and manager, who is curious about what happens inside networking boxes and wants to obtain a solid knowledge of networking.

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

Archive Index Issue Table of Contents

Advanced search

Customizing Vim

Dan Puckett

Issue #72, April 2000

Some great customizations to Vim's default behavior—make Vim work for you.

Vim is an editor designed to work like that most venerable of UNIX editors, vi. Vim doesn't just clone vi; it extends vi with features like multi-level undo, a graphical interface (if you want it), windows, block operations, on-line help and syntax coloring.

Along with the new features, Vim 5.5 (the current version as I write) has 196 options you can set. Practically any behavior you might have found obnoxious in plain vi can be configured to your liking in Vim. To download or get more information on Vim, see the Vim home page at http://www.vim.org/. Within Vim, you can view the on-line help at any time by pressing **ESC**, typing **:help** and pressing **ENTER**.

I'll admit that the thought of trudging through 196 options on the off chance that one or two will do what I want might seem a bit daunting, so here are several of my favorite Vim customizations just to get you started. These customizations have saved me much frustration and helped make a regular Vim user out of me.

Saving Your Customizations

Before I talk about specific Vim customizations, however, let me explain how to save your customizations so they are loaded each time you start Vim. When you first start using Vim, it will be 100% compatible with vi. You won't notice any of Vim's fancy features until you activate them.

This behavior is nice: it allows system administrators to replace /bin/vi with a link to Vim without their users rising up against them screaming, “vi is broken. Fix it!” In fact, some people have used Vim for years this way without realizing

they were using anything fancier than vi. But strict vi emulation can confuse people who expect to see all of Vim's bells and whistles right from the start.

Luckily, it's easy to convince Vim that we know we're actually in Vim and not in vi. Vim customizations are stored in a file called .vimrc in your home directory. If Vim sees that you have a .vimrc file—even if that file is empty—Vim will turn off vi-compatibility mode, which will configure Vim as Vim, rather than vi.

If you don't have a .vimrc file, but you do have an .exrc file that you have used to customize your vi sessions in the past, execute the command

```
mv ~/.exrc ~/.vimrc
```

to rename your .exrc file to .vimrc.

If you have neither a .exrc file nor a .vimrc file, execute the command

```
touch ~/.vimrc
```

to create an empty .vimrc file.

You're now ready to begin configuring Vim in earnest. You can add commands to your .vimrc file in the same way you would add them to your .exrc file. That is, if you tried Vim's incremental searching feature (which I'll describe shortly) by pressing the **ESC** key and entering the command

```
:set incsearch
```

and decided you wanted to make incremental searching the default behavior for future Vim sessions, you could do it by putting the line

```
set incsearch
```

into your .vimrc file on a line by itself. Note the lack of a leading colon.

Finding it Fast: incsearch

Suppose you have the following text file to edit:

```
In Xanadu did Kubla Khan
A stately pleasure-dome decree:
Where Alph, the sacred river, ran
Through caverns measureless to man
Down to a sunless sea.
```

Your cursor is on the **I** in the first line. You need to get to the first occurrence of the word “measureless”. How do you do it?

One way is to press **/** to put Vim into search mode, type in “measureless”, and press **ENTER**. Vim will find the first “measureless” after the current cursor position and leave your cursor on the **m**. Easy, in principle, that is. I'm not such a great typist. When I try to search forward for the word “measureless”, I'm just as likely to misspell it as not. And if I misspell it as “measurless”, I won't realize my mistake until I press **ENTER** and Vim returns “Pattern not found: measurless”.

I could increase my chances of typing the search pattern correctly by searching for a substring of “measureless”. For example, if I search for “measu”, I have fewer characters to type, which means fewer ways I can mistype my search pattern. However, that means I have to guess how many characters will specify a unique substring of the word I want to find. If I don't type in enough for my search pattern, I'll end up in the wrong location. For example, if *search* for “me”, I'll end up in “pleasure-dome” on line two rather than where I want to be, which is on line four. I'd then have to search again by pressing **n**.

Vim's incremental search feature can help with both of these problems. To try it out, press the **ESC** key to enter command mode, then type

```
:set incsearch
```

and press **ENTER**.

Incremental searching means that as you enter your search pattern, Vim will show you the next match as you type each letter. So when you start your search for “measureless” by pressing **m**, Vim will immediately search forward for the first **m** in the file following the current cursor position. In this case, it's the **m** in “pleasure-dome” on line two. Vim will then highlight in the text the pattern it has matched so far for you. Since “pleasure-dome” isn't where you wanted to go, you need to type more letters in your search pattern. When you press **e**, “pleasure-dome” still matches the substring **me**, so Vim will highlight the “me” in “pleasure-dome” and wait for more input. When you press **a**, “pleasure-dome” no longer matches the substring **mea**, so Vim will highlight the next match for **mea**, which is “measureless” on line four. Jackpot! Since that's the word you are looking for, press **ENTER**, and Vim will leave your cursor on the **m** in “measureless”.

With incremental searching, you always know what the results of your search will be, because the results are highlighted on your screen at all times. If you misspell your search pattern, Vim will no longer show you a highlighted match for your search pattern. When your highlighted match string disappears from the screen, you know immediately that you should back up by using the **BACKSPACE** key, and fix your search pattern. If you change your mind about

what you wish to search for, you can press the **ESCAPE** key, and Vim will return the cursor to its previous location.

Even Better Searching: ignorecase and smartcase

Programmers often don't capitalize code consistently. I'm no exception here. From one program to another—and sometimes even, to my shame, within the same program—my capitalization scheme changes.

“Let's see, was that subroutine named “CrashAndBurn”, “CRASHANDBURN”, “crashandburn” or “Crashandburn”?” If your editor is too picky about distinguishing upper-case from lower-case letters in its search patterns, you'll have a hard time matching the string. On the other hand, sometimes case is significant, and you do want to find “CrashAndBurn” and not “crashandburn”. What to do?

By default, both vi and Vim won't match anywhere in the text where the capitalization isn't exactly the same as the search pattern you entered; however, we can change this default behavior. Vim has a couple of options that, when used together, can take the pain out of upper/lower-case confusion. You can try these options by pressing the **ESCAPE** key, then typing the following two commands, pressing **ENTER** after each one:

```
:set ignorecase
:set smartcase
```

The ignorecase option is supported in vi as well as in Vim. It entirely disregards upper- and lower-case distinctions in search patterns. With ignorecase set, a search for the pattern “crashandburn” will match “CrAsHaNdBuRn” and “crashANDburn” as well as “crashandburn” in the text.

This is an improvement over the default behavior in some cases, but what if I really do want to search based on case distinctions? Will I have to set and unset ignorecase each time I want to search a different way?

In vi, the answer, unfortunately, is yes. Vim is a little more subtle, though, in that it offers the smartcase option as well. If both ignorecase and smartcase are set, Vim will ignore the case of the search only if the search pattern is all in lower-case. But if there are any upper-case characters in the search pattern, Vim will assume you really want to do a case-sensitive search and will do its matching accordingly.

For example, with both ignorecase and smartcase turned on, Vim will match “crashandburn” with both “CrashAndBurn” and “crashandburn”. If you enter “CrashAndBurn” as your search pattern, however, Vim will only match the string “CrashAndBurn” in the text. It won't match “crashaNDBUrn”.

In practice, this combination of options works out to be a good compromise, letting you balance case-sensitive and case-insensitive searches nicely without having to set or unset an option to do them.

Keep Some Context: scrolloff

When I'm editing a program or document, I like to have a little context around my work by keeping the line of text I'm working on a couple of lines away from the edge of the window at all times.

In vi, I would maintain this bit of context by scrolling a few lines either above or below the line I wished to edit, then moving back to my destination and doing my editing. It wasn't great, but it was better than typing blind, which is how I felt whenever I worked on the first or last line of the screen.

Luckily, Vim can maintain some context for you automatically through the use of the scrolloff option. You can try setting this option by pressing the **ESC** key and entering

```
:set scrolloff=2
```

The **2** means I want at least two lines of context visible around the cursor at all times. You can set this to any number you like. Vim will scroll your file so that your cursor will never be closer to the top and bottom edge of the screen than the number of lines you specify.

Vim won't always be able to honor your scrolloff specification. If you're near the bottom or top of the file, there may not be enough lines left between your cursor and the file's beginning or end to give you the context you asked for. It will do the best it can, though.

I recommend the scrolloff feature highly. It's been a great help to me.

File Name Completion: wildmode

I hate typing file names. Why should I have to type out a file name like “thelongestfilenameintheworld.html” if the starting characters “thelong” will uniquely identify it from all other files in the current subdirectory? I also have the habit of wanting to edit a file deep within an unfamiliar directory structure.

Luckily, Vim has file name completion. File name completion lets you enter a partial file name into Vim, then press the **TAB** key to have Vim search for a file or directory name that could complete it. If Vim finds exactly one file or directory that matches, it fills in the rest of the name. If Vim can't find any match, it beeps.

What if Vim finds more than one file or directory name that matches? You can specify what Vim does next in this case by setting the wildmode option. The default setting for wildmode is “full”. When wildmode is set like this, the first time you press **TAB**, Vim will fill in one of the files or directory names that match what you have typed so far. If you hit **TAB** again, Vim will show you another file that completes your match. As you keep pressing **TAB**, Vim will go through all the possible completions. When it runs out, the next time you press **TAB**, Vim will show you the original incomplete string you entered. Now you're back where you started. If you press **TAB** again, Vim will show you the first match again.

While this is good, I prefer my file name completion to work a little differently. Here's how I like to have wildmode set:

```
:set wildmode=longest,list
```

Setting wildmode this way makes Vim act as follows. When I enter part of a file name and press **TAB**, Vim completes my file name to the longest common string among the alternatives. It then waits for me to do one of the following: press **ENTER** to accept that as the file name, keep typing the file name from that place, press **ESC** to cancel the command, or press **TAB** again. The second time I press **TAB**, Vim will list all possible files that could complete my partial file or directory name.

Don't like either of the file completion methods I listed above? Not to worry: wildmode has many different options. For details, enter

```
:help wildmode
```

and Vim will show you every possible option.

Conclusion

Enjoy customizing Vim. If you take one step at a time, you'll find that using Vim becomes more and more pleasant as time goes by. I think the more you make Vim work your way rather than its default way, the more you'll come to like it.

More Information

email: puckett@acm.org

Dan Puckett supports configuration management software for a rather large telecommunications firm. He is a writer and actor. He welcomes your comments at puckett@acm.org.

Archive Index Issue Table of Contents

Advanced search

Converting from SCO Xenix to Linux

Fred Treasure

Issue #72, April 2000

A computer consultant's experience in converting the SBT accounting system to Linux for the Maxwell House Department Stores.

There are many customized versions of SBT's accounting system programs because SBT (http://www.sbt.com/) has been supplying source code with the multi-user versions of their software for many years. The system I upgraded from SCO Xenix to Caldera OpenLinux used SBT's Accounts Receivable/ Inventory program as the basis for a point-of-sale system. Both the original SBT programs and the add-ons were written in SCO Foxbase plus, one of the dBase-compatible languages. My client, Steve Maxwell, had been using the program for about ten years to run his department stores, and was pleased with its performance and stability. Unfortunately, when I tested how well this software worked with dates in the year 2000, it failed. SCO Foxbase did not work correctly after 1999.

While I was looking into the available solutions to the problem, I found, in addition to an upgrade from SCO, a couple of dBase-compatible languages also available for Linux. I liked a product called FlagShip from Multisoft (sold in the U.S. by Linux Mall, http://www.linuxmall.com/), because my initial tests indicated it would compile the original SBT source code with only a few changes, which I'll detail later, and it produced very fast code. It was also the least expensive solution, at less than $600. FlagShip compiles the source code in a two-step process that converts the original dBase source into C code, which is then compiled by Linux's C compiler into a native executable.

From the client's point of view, one of the major advantages of the upgrade to Linux was it allowed him to use much of his existing hardware, including the many terminals, receipt printers, bar code scanners and so on, as well as his main computers. The only hardware we had to replace was a 60MB tape drive and a multiport serial card. Both were no longer manufactured and were not supported by Linux. We upgraded the multiport serial board to a Cyclades

Cyclom Y board, and upgraded the tape drive to a Hewlett Packard IDE 5GB unit with BRU backup software. The ability to recycle major components saved tens of thousands of dollars. By using the original programs, we also avoided training costs.

Since this was my first Xenix-to-Linux conversion, I was fairly concerned I would run into an unforeseen problem that would make the project fail. In this regard, I was fortunate to have a very experienced client. Steve Maxwell had been involved in the development of the original Xenix version of the programs, and had been maintaining the system without outside support. I felt much better having a truly smart client around, because I ran into trouble right away: I couldn't get the information on the Xenix system over to the Linux system. Although Linux can read Xenix file systems, I was not able to get Linux to read the Xenix hard drive that held the original source code and data. I was able to get the information off of the Xenix system using a program Steve called **Term**, which has both Xenix and DOS versions. I wound up with all needed files on the DOS partition of my Linux development system. This turned out to be quite useful in that it allowed me to use all the DOS development tools, in addition to the Linux tools, and produced code I could debug in either environment.

I started the conversion by doing the Y2K work. I converted anything that had to do with dates from the original format of 8 bytes of character data (e.g., 01/01/90) to dBase's date type. This is the approach used by SBT for its DOS and Windows products. I used the **SET CENTURY ON** command to enable four-digit years, then made the necessary changes to convert the date where needed. This turned out to be relatively easy.

In the next step of the conversion, I built a stripped-down copy of the department store's system, consisting of a main computer and a single point-of-sale workstation (a dumb terminal, bar code scanner and receipt printer). I also set up some system printers in the configuration used by the store.

None of the available Linux terminal types worked, and when I logged out from the terminal, I didn't get a login prompt again. This was something I hadn't expected and was a pain to resolve. I eventually produced a custom **terminfo** entry by reverse engineering the Xenix termcap settings. The login problem was solved by using **mingetty** instead of the standard **getty** program. After resolving these problems I had a working single-terminal version of the system.

The way SCO Foxplus handles a multi-user file is different than the way FlagShip handles it. Basically, I stripped out the Foxplus code and added the FlagShip code as needed, an easy process, but a bit tedious since I had to add a block of FlagShip code at each point in the program the **USE** command appeared. In about a month, I had finished all conversions and installed the

Linux hard drive (which I'd done the development on) as the main store computer. I kept the Xenix hard drive handy, in case there was some more trouble in the upgrade. Amazingly enough, the Linux version worked on the first try. After that, I made minor changes to the code to optimize the way the printers worked, but the Linux system was up and running and I never needed to go back to Xenix.

The original point-of-sale programs supported multiple stores by updating a set of master files through a complicated operation involving transaction files created at each store and a batch update process. I was able to simplify this by using Linux's powerful communication features to connect the branch store directly to the main store via the Internet.

I obtained two dedicated IP addresses from Gilanet, the local ISP that serves the two towns where the stores are located, and configured the automatic dialer program to bring up the links during store hours and shut them down when not needed. Bill Stites, the system administrator from Gilanet, provided great support in getting the Internet connections working. Bill uses Caldera OpenLinux internally for some of his Gilanet servers.

When we started this project, Steve Maxwell and I had wanted to complete the upgrade prior to the busy Christmas season. As it turned out, even with the problems, I was able to complete the work in about half the expected time. The system runs beautifully.

email: fred@nfo.edu

Fred Treasure can be reached via e-mail at fred@nfo.edu.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Large-Scale Linux Configuration Management

Paul Anderson

Issue #72, April 2000

Mr. Anderson describes some general principles and techniques for installing and maintaining configurations on a large number of hosts and describes in detail the local configuration system at Edinburgh University.

The difficulty of installing and setting up Linux is often mentioned as one of the reasons it is not more widely used. People usually assume that editing the traditional UNIX configuration files is more difficult than using the graphical interfaces provided by operating systems like Microsoft Windows. For a novice user with a single machine, this may be true, and most commercial UNIX vendors now supply GUI-based tools for at least some aspects of system configuration. Under Linux, projects like COAS (see Resources 1) and the Red Hat distribution are starting to cater to this need.

For a large installation with tens or hundreds of machines, the GUI approach does not work—entering individual configuration data for 200 machines is simply not practical. As well as the ability to install large numbers of machines, big sites usually need more control over the configuration; for example, they might need to install new machines with a configuration which is guaranteed to be identical to an existing one. Machines are also likely to need periodical reconfiguring as their use changes, or simply to keep up to date with the latest software and patches.

To do this effectively requires a good deal of automation, and large UNIX sites have been developing their own tools for many years (see Resources 2). The flexibility and accessibility of UNIX configuration files makes Linux particularly suitable for automation, and those sites attempting to install and manage large numbers of NT systems are often likely to find the process more difficult (see Resources 3).

The Division of Informatics at Edinburgh University has over 500 UNIX machines, with a wide variety of different configurations. Most of them are

installed and maintained automatically using the *LCFG* (Local ConFiGuration) system, originally developed several years ago (see Resources 4). Both client and server configurations can be easily reproduced to replace failed machines or to create tens of identical systems for a new laboratory. Reconfiguration is thus a continuous process; for example, machines adjust every night to ensure they are carrying the latest versions of the required software. Linux (we use a version of the Red Hat distribution) has proven itself well-suited to this environment, and it has recently overtaken Solaris to become the most popular desktop system, both for staff use and student laboratories.

Make-Up of a Good Configuration System

An automatic configuration system should be able to build working machines from scratch with no manual intervention. This includes configuration of the basic operating system (disk partitions, network adaptors), loading of required software, and configuration of application-specific services such as web servers. This allows failed machines to be recreated quickly, using replacement hardware, and new machines to be installed efficiently, even by junior staff. As a side effect, it also avoids the need for backups of any system partition.

The set of configuration information that drives this build process defines the *personality* of an individual machine, and it is extremely useful if this *specification* is available in an explicit form (such as a plaintext file or a database). Machines can then be *cloned* simply by copying their specification and applying the automatic build. This is important for installing multiple similar machines, such as in a student laboratory. The master copy of the specification should be held remotely from the machine, so that it is available even when the machine is down. This allows programs to automatically verify individual configurations and even the relationships between machines, such as ensuring every client's specified DNS server is actually configured to run a name daemon. The specification can also be generated from higher-level descriptions of a machine's function. An inheritance model is very useful, since many machine configurations can be conveniently described as small variations of a generic configuration for a particular class.

Traditional configuration systems are often *static*, in the sense that the configuration is applied only at the time the machine is installed. Most vendor-supplied installation processes fall into this category, as do systems based on cloning by copying disk images. If subsequent changes to the configuration have to be applied manually, the configuration is almost certain to “rot”, and it is impossible to be confident that all machines are correctly configured. Obvious misconfigurations simply result in users having malfunctioning machines. More subtle misconfigurations may go unnoticed and pose serious security problems, for example. Even though a fully *dynamic* system is not

practical, an ideal system will continually adjust the configuration to conform to the specification. Some parameters can be changed immediately to track a change in the specification; some, such as a network address, may be changed only when the machine reboots; and others, such as a disk partitioning, may require a complete rebuild.

If a configuration system is incomplete and manual intervention is necessary, many of the benefits are lost. However, constructing a comprehensive system to cover every conceivable parameter is clearly impractical. The key problem is trying to create an extensible framework flexible enough to allow new parameters and *components* to be incorporated with little effort. An individual instance of the system can then evolve at a particular site to suit the local requirements. If it is going to be extended on demand by working administrators, the framework needs to be extremely lightweight and comprehensible in a short amount of time. It must be easy to create components in a familiar language, and to interface them to new subsystems which require configuration. Open-source software is an advantage, since it is often easy to base a new extension on one that already exists.

The LCFG Framework

Before the introduction of LCFG, we were configuring machines using a typical range of techniques, including vendor installs and disk copying (cloning). These were followed by the application of a monolithic script which applied assorted "tweaks" for all the different configuration variations. This met with virtually none of the requirements listed above and was a nightmare to manage.

The available alternatives ranged from large commercial systems (too expensive and probably too inflexible) to systems developed at individual sites for their own use (often not much of an improvement over our existing process). More recently, interesting tools such as COAS and the GNU **cfengine** (see Resources 5) have appeared, but we are still not aware of any comparable system which addresses quite the same set of requirements as LCFG.

Given limited development resources, we attempted to design an initial system as a number of independent subsystems, intending to use temporary implementations for some of the ones where we could leverage existing technology:

- *Resource Repository*: design a standard syntax for representing resources (individual configuration parameters). These would be stored in a central place where they could be analysed and processed as well as distributed to individual machines.

- *Resource Compiler*: preprocess the resources so that we could create configurations by inheritance and avoid specifying large numbers of low-level resources explicitly.
- *Distribution Mechanism*: distribute the master copy of the resources to clients on demand in a robust way.
- *Component Framework*: provide a framework which allows components to be easily written for configuring new subsystems and services, using the resources from the repository.
- *Core Components*: implement a number of core components, including basic OS installation and the standard daemons. We wanted some of these to act as exemplars to make it as easy as possible for other people to create new components.

The Resources

Items of configuration data are represented as key,value pairs, in a way similar to X resources. The key consists of three parts: the hostname, the component and the attribute. For example, the nameserver (cul) for the host wyrgly is configured by the DNS component:

```
wyrgly.dns.servers: cul.dcs.ed.ac.uk
```

Notice that this specification is a rather abstract representation, not directly tied to the form in which the configuration is actually required by the machine, in this case, as a line in the resolv.conf file. This allows the same representation to be used for different platforms, and it permits high-level programs to analyse and generate the resources easily . The LCFG components on each machine are responsible for translating these resources into the appropriate form for the particular platform. COAS uses a similar representation for configuration parameters.

The resources are currently stored in simple text files, with one file per host. This collection of files forms the *repository*. We intend to provide a special-purpose language for specifying these resources; it would support inheritance, default configurations, validation and some concept of higher-level specifications. However, we are currently using a “temporary” solution based on the C preprocessor, followed by a short Perl script to preprocess the resources. The C preprocessor provides file inclusion and macros, which can be used for primitive inheritance. The Perl script allows inherited resources to be modified with regular expressions. Wild cards are also supported to provide default values.

In practice, most machines have very short resource files which simply inherit some standard templates. Machines can be cloned simply by copying these

resource files. Often, a few resources are overridden to provide slight variations. For example:

```
#include <generic_client.h>
#include <linux.h>
#include <portable.h>
amd.localhome:  paul
auth.users:     paul
```

The name of the host is not necessary in the resource keys, because this is generated from the name of the resource file.

Resources are currently distributed to clients using NIS (Sun's Network Information System). This is another “temporary” solution which is far from ideal; we hope to replace it in the near future.

The Component Framework

A number of components on each machine take the resources from the repository and implement the specified configuration in whatever way is appropriate for that particular platform. The components are currently implemented as shell scripts which take a standard set of *method* arguments, rather like the rc.d startup scripts under Red Hat Linux:

- **START**: executed when the system boots.
- **STOP**: executed when the system shuts down.
- **RUN**: executed periodically (from **cron**).

A client-server program (**om**) also allows methods to be executed on demand on multiple remote machines. Components may have other arbitrary methods in addition to the standard ones.

Different types of components will perform different actions at different times. Typically, a daemon might be started at boot time, reloaded periodically, and stopped at shutdown. Some components however, might simply perform a reconfiguration at boot time, or start only in response to the *RUN* method (for example, a backup system).

Component scripts normally inherit a set of subroutines from a *generic* component. This provides default methods and various utility procedures for operations such as resource retrieval. This makes simple components easy to write, and scripts are frequently quite short.

Some Important Components

A typical host runs 20 to 30 components, controlling subsystems such as web servers, printers, NIS services, NFS configuration and various other daemons. Two components are worth mentioning in more detail.

The **boot** component is the only one run directly from the system startup files. This uses resources to determine which other components to start. The set of services running on a particular machine is therefore controlled by the boot resources.

The **update** component normally runs nightly, as well as at boot time. This uses the extremely useful **updaterpms** program which compares the RPMs installed on a machine with those specified via the resources. RPMs are automatically installed or deleted to synchronise the state of the machine with the specification. This means that all machines in the same class are always guaranteed to have identical sets of up-to-date packages. Changing an inherited resource file will automatically reconfigure the RPMs carried by all machines in the class.

Machine Installation

As much configuration as possible is performed dynamically by the various components. However, some configuration, such as disk partitioning, must be hard-wired at installation time. New machines are booted using an installation floppy, which mounts a root file system from the network, a CD or Zip drive. The boot process runs a special **install** component which determines all necessary install-time parameters by interpreting the machine's *install* resources. A very minimal template is installed on the new system and the **update** component is used to load the initial set of RPMs.

This supports completely unattended builds of new machines, as well as rebuilds of existing machines. If there is any doubt about the integrity of a system, it is normal for us to simply rebuild it from scratch.

Problems and Future Plans

The concept of an open, lightweight framework has been very important; many people have contributed components so that virtually everything which varies between our machines is now handled by LCFG. This has made the system very successful; however, much of the implementation is still based on technologies originally intended to be temporary. We are currently planning to expand the

use of LCFG beyond our own department and this is motivating a redesign of some of the subsystems, although the basic architecture will remain the same:

- We hope to implement a new syntax for specifying the resources, together with a special-purpose resource compiler.
- We hope to replace the NIS distribution with something simpler which is available earlier in the boot sequence.
- We would like to re-implement the components in Perl, using Perl inheritance to provide generic operations.

Other items on the wish list include caching support for portables and secure signing of resources.

Resources

Acknowledgements

email: paul@dcs.ed.ac.uk

Paul Anderson is a Senior Computing Officer with the Division of Informatics at Edinburgh University. He has been involved with UNIX systems administration for 15 years. Further information is available from www.dcs.ed.ac.uk/~paul, and comments by e-mail are welcome at paul@dcs.ed.ac.uk.

Archive Index Issue Table of Contents

Advanced search

The Linux Scheduler

Moshe Bar

Issue #72, April 2000

A look at how the kernel schedules tasks for both uni-processor and multi-processor machines.

Last month, we started a new series on Linux kernel internals. In that first part, we looked at how Linux manages processes and why in many ways Linux is better at creating and maintaining processes than many commercial UNIX systems.

This time, we dwell a bit on the subject of scheduling. Much to my surprise, here again, Linux goes the unorthodox way and disregards conventional wisdom in kernel theory. The results are excellent. Let's see how.

Scheduling Classes

In Linux 2.2.*x* there are three classes of processes, as can be seen from the data definition for the scheduler (from linux/include/linux/sched.h):

```
/* Scheduling Policies
*/
#define SCHED_OTHER  0
#define SCHED_FIFO   1
#define SCHED_RR     2
```

SCHED_OTHER tasks are the normal user tasks (default).

Tasks running in **SCHED_FIFO** will *never* be preempted. They will leave the CPU *only* for waiting sync kernel events or if an explicit sleep or reschedule has been requested from user space.

Tasks running in SCHED_RR are real time (RT), but they will leave the CPU if there is another real-time task in the run queue. So the CPU power will be distributed between all **SCHED_RR** tasks. If at least one RT task is running, no other **SCHED_OTHER** task will be allowed to run in any CPU. Each RT task has an

rt_priority so the **SCHED_RR** class will be allowed to distribute the CPU power between all the SCHED_RR tasks at will. The **rt_priority** of the **SCHED_RR** class works exactly as the normal priority field for of the **SCHED_OTHER** (default) class.

Only the root user can change the class of the current task via the **sched_setscheduler** syscall.

One of the tasks of a kernel is to make sure the system remains firmly under its control even in situations of misbehaving programs. One such misbehaving program might **fork** too many processes too quickly. Thus, the kernel becomes so busy with itself that it cannot cater to its other responsibilities. I found out that Linux has no limit to how fast user-land programs can spawn children. HP-UX, Solaris and AIX have a limit of one fork per processor tick (called a jiffie under Linux). The patch in Listing 1 (see Resources) will allow a maximum of one fork per jiffie (one jiffie is usually 1/100 second, except on the Alpha architecture where it is 1/1024).

Tangled in the Threads

Threads are necessary to allow your process to make use of multiple processors. Linux doesn't really make any distinction between a process and a thread from a memory management and scheduling point of view. Some operating systems, like Solaris, manage threads within the user process by means of a thread scheduling library. The kernel sees only the process and doesn't know which thread, if any, is actually executing inside the user process. This saves the kernel from managing lists with thousands of entries for each thread for each process.

Obviously, the threads emulated on the top of one single user process won't be allowed to run concurrently on SMP, so the user-space approach won't scale very well on an SMP machine. Threading is strictly necessary only when all threads will be CPU-bound and not mainly I/O-oriented. If all the threads are CPU-bound, you definitely want to be able to scale for SMP.

Using threads only to wait for events is overkill. On the other hand, having threads sleeping is a waste of resources and performance. Almost all subsystems in Linux (such as TCP/IP) offer async event registration. Using async event via the **SIGIO** signal is similar to IRQ-driven handling.

With the user-space approach, you will at least avoid the TLB (translation lookaside buffer) flushing, as all the threads will share the same address space.

The advantage of having threads managed in user space through the threads library is the kernel will spend the scheduling CPU cost in user space. It is true

that in user space, you may choose to implement a very fast round-robin scheduler that may cut down the scheduling cost, compared to the clever (but more expensive, in terms of execution path) Linux scheduler.

Speaking of SMP, as of Linux 2.4 I found there is no way to declare the processor affinity of any given user-space process.

The scheduler could keep track of the CPU affinity declaration of a process, or it could just determine a preferred CPU for a process by itself. The other day, together with Andrea Arcangeli of Italy, I designed the simple kernel patch in Listing 2 (see Resources) that implements processor affinity. Notice that processor affinity makes the most sense when other processes are excluded from running on this CPU. A better way to implement this patch would be to have the system administrator set affinity with an external call like **nice**.

The 2.2.*x* SMP kernel scheduler has some bugs that sometimes make it less effective than the UP (UniProcessor) scheduler. Nevertheless, Andrea fixed all such bugs and rewrote the heuristics from scratch and the SMP scheduler gives an impressive SMP improvement under load. The SMP changes are just in the 2.3.*x* kernels, and I plan to integrate it in 2.2.*x* also. You can get Andrea's patch at ftp.suse.com/pub/people/andrea/kernel-patches/my-2.2.12/SMP-scheduler-2_2_11-E. That patch can be used against 2.2.11 and 2.2.12 and speeds up both kernels on SMP systems. The patch was merged into 2.3.15, but not yet in 2.2.*x* because it's a performance issue only and not a true bug fix.

The SMP scheduler heuristic mechanism works as a function of (not in any particular order):

- the idle CPUs
- the last CPU where the **wakenup** task was running
- the memory management of the task (for optimising kernel-threads reschedule)
- the “goodness” of the tasks running on the busy CPUs
- the time necessary to invalidate the L2 cache on the running CPU (**cacheflush_time**)
- the average time slice (**avg_slice**) of the wakenup task (how much time the task runs before returning to sleep)

The algorithm collects the above data and chooses the best CPU on which to reschedule the wakenup task.

There are two paths involved in the Linux scheduler behavior:

- **schedule**: the running/current task is a SCHED_OTHER task that expired its time slice (so the kernel runs a schedule while returning from the timer IRQ for switching to the next running task).
- **reschedule_idle**: a task got a wakeup (usually from an IRQ), and so we try to reschedule such wakenup task in the best CPU by invoking a schedule on it (it's a kind of controlled schedule).

Both paths share the **goodness** function. The goodness function can be considered the core of the SMP scheduler. It calculates the “goodness” of a task as a function of the following:

- the task currently running
- the task that wants to run
- the current CPU

The source for the goodness function can be found in Listing 3 (see Resources).

Listing 3

A plain schedule only works based on goodness. As you can see in Listing 3, a plain schedule is SMP-aware. The goodness of the potential next task increases if its last CPU is the current CPU.

Nevertheless, reschedule_idle is far more critical for CPU affinity and scheduler latencies; for example, if you comment out reschedule_idle, the scheduler latency will become infinite. Also, reschedule_idle takes care of the cache-flush time and the task average time slice, and this is the truly interesting part of the SMP scheduler. In UP, reschedule_idle is not as interesting as the SMP version. Listing 4 (see Resources) is the reschedule_idle implementation taken from 2.3.26 (soon 2.4).

The final objective of reschedule_idle is to call a schedule on a CPU in order to reschedule the wakenup task in it. We use goodness in reschedule_idle because we want to predict the effect of the future schedule that we'll send to that CPU. By predicting the effect of the future schedule, we can choose the best CPU to reschedule at wakeup time. This, of course, saves us the trouble of executing on a CPU without the proper TLB settings. If the CPU to reschedule is not the current one, we send a reschedule event via inter-CPU message passing (SMP-IPI interrupt on i386).

To make it very clear: the goodness function is the core of the Linux scheduler and is SMP aware, while reschedule_idle is the core of the clever SMP heuristics.

Kernel Preemption and User Preemption

Linux can only do user preemption. Linus Torvalds, it seems, doesn't believe in kernel preemption. That's not as bad as it may seem; all is fine for semaphores. Critical sections protected by semaphores can be preempted at any time, as every contention will end in a schedule and there can't be any deadlock. However, critical sections protected by fast spin locks or by hand locks cannot be preempted unless we block the timer IRQ. So all spin locks should be IRQ-safe. Also, by avoiding kernel preemption, the kernel becomes more robust and simpler, since a lot of complicated code can be saved this way.

By the way, there are tools to monitor the scheduler latency in order to allow the interested hacker to catch potential code sections that need conditional schedules.

Implications for Linux

Linux, due to its GPL nature, allows us to do things faster than others, because you can adapt and recompile your own kernel instead of using a standard fit-all kernel. For example, in some popular proprietary operating systems such as Solaris 7, many code sections have been packaged within **spin_lock** and **spin_unlock** to make the same code work well in both UP and SMP systems. While vendors of these commercial operating systems tout this as a clear advantage for heavy SMP systems, these locks actually bog down UP systems and simple SMP machines, because the same binary driver must work on both SMP and UP kernels. A **spin_unlock** is one locked ASM instruction:

```
#define spin_unlock_string \
        "lock ; btrl $0,%0"
```

and calling such clear-bit instruction through a function pointer is complete *overkill*.

Linux, on the other hand, has in-line code sections for UP and SMP systems, adapting to the machine it is running on. Therefore, if only a UniProcessor system is hosting the kernel, no time is lost locking a code section that doesn't need it.

Last but not least, as we saw above, the goodness function makes the SMP scheduler in Linux very clever. Having a clever SMP scheduler is critical for performance. If the scheduler is not SMP-aware, OS theory teaches us that the performance on an SMP machine can be even worse than on a UP machine.

(This was happening in 2.2.x *without the new heuristics, but has now been fixed.)*

That's it for this month. I hope you gained some deeper understanding of how Linux schedules tasks on UP and SMP systems.

Resources

Moshe Bar (moshe@moelabs.com) is an Israeli system administrator and OS researcher, who started learning UNIX on a PDP-11 with AT&T UNIX Release 6 back in 1981. He holds an M.Sc. in computer science. He has written a book, Linux Kernel Internals, to be published by McGraw-Hill this year.

Archive Index Issue Table of Contents

Advanced search

Cookie Cutters, Munchers and Crunchers

Marcel Gagné

Issue #72, April 2000

Some clues for protecting your privacy on the Internet.

Bonjour, and welcome once again to my restaurant, “Chez Marcel”. Please, sit down. This month, the special is Internets and Intranets. The world, you see, has become a very small place, with every other person or business spinning a web site. Even Chez Marcel is considering opening the restaurant to the world, but François feels it would spoil the intimate nature of the place.

François! What are you doing sitting around? Wine for our guests. Vite! Vite!

While François is getting your Chablis, I would like to tell you about the features on tonight's menu—cookie cutters, munchers and crunchers. Now, I am well aware that my fellow columnist, Reuven Lerner, once provided you with a recipe for making cookies (see “At the Forge” in *LJ* issue 45). What I want to do today is share recipes for destroying or deleting cookies. After all, too many cookies can be quite fattening, non?

Before sampling our first item, it might be interesting to delve into a history and explanation of the cookie. Internet cookies are sometimes called “magic cookies” and should in no way be confused with “magic mushrooms”, since Internet cookies are not edible. Cookies are simply small text files transmitted to your browser (or system) when you visit a web site. Cookies do nothing on your system, other than sit there. This is because, as I mentioned before you finished off that last glass of wine, *cookies are just text*--no code or programs.

The whole idea behind cookies was that a server would give you a cookie as a marker to indicate where you had previously visited. That cookie might store a user name and password to access a particular web site. (Sites will often—but not always—have this information encoded so that somebody viewing your cookie file would see only what appears to be junk text. Have a look at your own cookie file.) When you next visit the site, the server would ask you whether

it had served you any cookies, and your browser would reply by sending the cookies from before. In this way, the web site would recognize you, directing you to your virtual table. You see, no web site, no matter how clever, has François to recognize you as you walk in.

Originally, cookies were left by a site and picked up by the same site, allowing for a kind of interactive session. For example, after a long night of shopping at your favorite web store, you decide to pack it in and come back in the morning. The next day, the store's web site requests the cookies it left with you and “knows” that you had three pounds of cheese, two jars of Dijon mustard and a nice Cabernet in your shopping cart. Cookies can be quite useful, as you see.

The problem is that cookies can also be shared within larger domains, such as advertising rings. Using these shared cookies, advertisers can build a profile of your likes and dislikes, tailoring and targeting advertising to you specifically. After a while, every site you visit that deals with certain banner advertisers knows your weakness for Brie, and in no time, you are broke. Your privacy has been compromised, so that others may offer to sell you what you cannot resist. Many people object to this method of building user profiles, and consider the use of cookies to be quite unethical. If you are concerned about cookies, consider the following items from our menu.

The easiest way of all is *simply to refuse all cookies* in the browser itself. If you are using Netscape, click on “Edit”, then choose “Preferences”. In the window that pops up, look to the left-hand menu. Click on “Advanced” and select the level at which you want to handle cookies. Notice that you can refuse cookies entirely, accept them all at face value, or request that you be warned before the browser accepts them. (See Figure 1.) You can also further specify this behavior based on whether the cookie originated on the server of the page being viewed. In other words, no cookies from a banner advertising server or any other hosts being linked to by the site you are visiting.

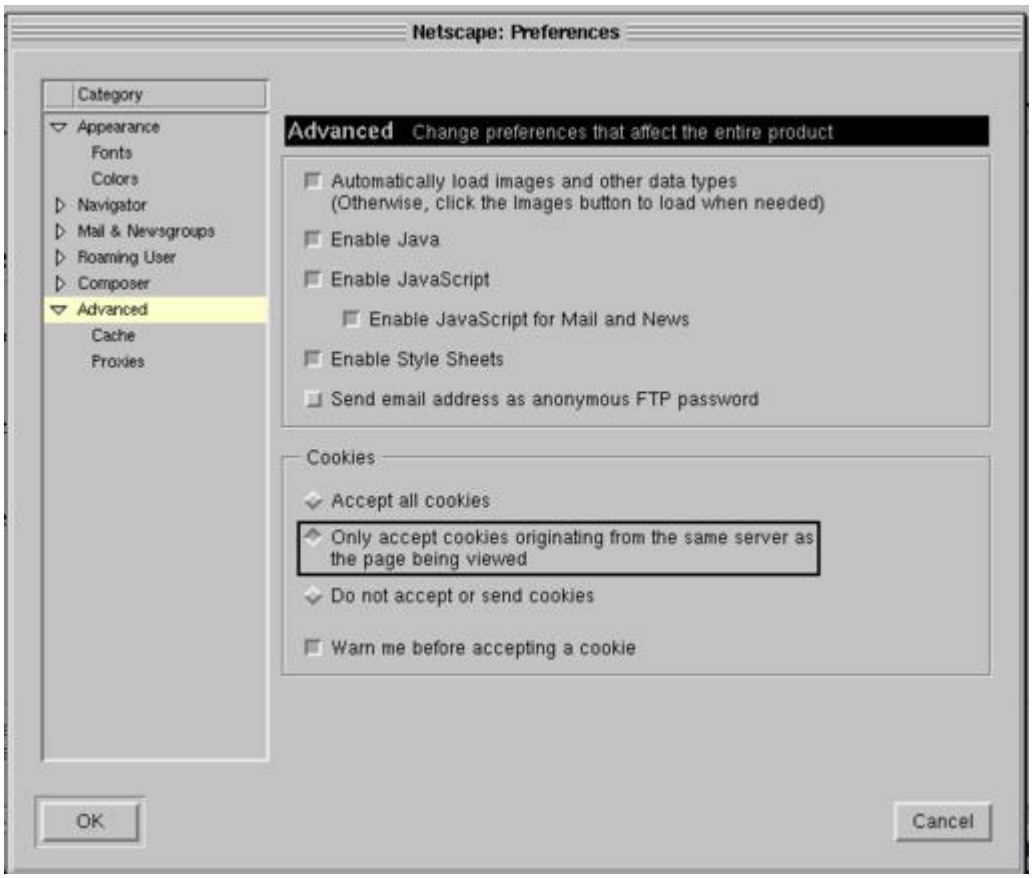


Figure 1. Netscape's Cookie Dialog

If you don't want to bother with selecting this cookie or that cookie, you may find the next items more to your taste. After all, clicking “OK” or “Cancel” over and over again when visiting a page that serves up dozens of cookies can be tiring. How do we avoid this?

You could do it the hard way. Each time you want to launch your browser, remove the cookies file from your user name's .netscape directory and start Netscape. Allow me to demonstrate, non?

```
rm -f ~/.netscape/cookies
```

The tilde character (~) is a way of referring to your home directory. The **$HOME** variable also works here. This simple command will do a nice job of cleaning things up. Then, just click on your Netscape icon, or start Netscape from the command line. You could also do what I did, and write a very simple shell script to do the job. Let's call it “netscape_clean”. It would look something like this:

```
#!/bin/bash
# This starts Netscape with a clean cookie file.
rm -f ~/.netscape/cookies
/usr/bin/netscape
```

Keep in mind that the path to the **netscape** executable may vary from system to system. On my system, it lives under /usr/bin.

As I mentioned earlier, cookies can be quite useful, so simply removing everything is not necessarily the way to go. Tech support web sites may not be able to process your queries, web stores and their shopping carts may simply not work, or content may be refused to you entirely. You may want to be more selective in what you destroy.

The recipe shown in Listing 1 requires a little Perl for added bite. You still need to run this outside of your current browser session. The difference here is you will be prompted, line by line, for cookies you wish to keep. If they appear to be what you want, simply answer with a capital “A” for “Accept” and continue on. The sharper eyes will note that to delete a cookie from the active file, you need only press **RETURN** or **ENTER**. Et voil<\#224>! A much leaner cookies file, filled with only nutritious information.

Listing 1

Of course, you do not have to do it this way, but my Perl script should serve as a great starting point for you to season to your tastes. One idea might be to add a check for the Netscape comment lines at the top of the file, so that you would not need to accept or delete these. They would simply be copied into the other file. Then again, perhaps not. They are simply comments, and my Netscape program re-adds them if they are missing.

After experimenting with the above recipe, I decided to strike out into the very heart of the Web itself in order to provide you, my esteemed guests, with an alternative recipe. My travels took me to LinuxBerg where a search on the word “cookie” brought me to Phil Darnowsky's “cookiecutter”, another Perl script that includes a “ban and allow” list. His script takes a different approach from my own creation, but the results are the same—no cookies you do not want.

Before I wrap up this month's column, might I suggest that your on-line privacy may be more important than simply deciding who can collect information on you via cookies. There are other ways to track your surfing habits and, to some degree, your identity, simply by asking your browser for information. For a demonstration of this, check out Privacy.Net's Anonymizer privacy analysis at http://privacy.net/anonymizer/. You may find it quite fascinating (and maybe a bit scary) to sample what a site can discover about you and the system you are running.

How does one block such information, you ask? One way is to run a proxy between your browser and the rest of the world. One example is the Internet Junkbusters Proxy, a program released under the GPL that is available for Linux and a number of other platforms.

You may also wish to completely block all your surfing from prying eyes by doing it behind a service provided for just such anonymity. This is where the Anonymizer comes into play. Using this company's service (they offer both a free and a premium service), you can enter the page you want to visit into the form provided and surf completely *incognito*.

These methods of securing your anonymity on the Web extend beyond the realm of the cookie, but the cookie remains a fascinating beast. If you want to know about all things cookie, you should pay Cookie Central a visit. This site contains a wealth of information about Internet cookies.

Well, mes amis, that terrible horloge on the wall tells me it is once again nearly closing time. Before you go, consider trying a fun little demonstration of the cookie—pay a visit to Privacy.Net's cookie page at privacy.net/cookies. I mention this one because, as part of the demonstration, they will "bake" your favourite cookie for you (in a virtual sense). Your chef admits to being a bit of a sucker for Fig Newtons, which was my choice. Your tastes may be different. Chocolate chip, perhaps?

Au revoir, mes amis. Remember, you are always welcome here at *Chez Marcel*. Bon Appétit!

Resources

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting Inc, a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy and edits TransVersions, a science fiction, fantasy and horror magazine. He loves Linux and all flavors of UNIX and will even admit it in public.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Designing Databases

Reuven M. Lerner

Issue #72, April 2000

Structuring tables can improve database performance-- here's how to do it.

Relational databases are becoming increasingly popular for web applications. This is generally a good thing, allowing us to focus on the way in which our data is structured, rather than the way it is stored on disk. Offloading data storage and retrieval tasks to a relational database server means our programs can be smaller and easier to maintain.

However, incorporating a database server on a web site is not a cure-all. The database might take care of many necessary tasks, but it cannot design your tables for you, nor determine the best way in which to work with them.

This month, we will look at the art of database design and how we can structure tables to improve performance. Getting the most out of a database is something of a black art, which is why good database administrators (DBAs) are always in high demand. But with a few simple techniques, we can overcome some of the most basic performance problems experienced by web programmers. We will design a database that can handle train schedules. In so doing, we will touch on a number of issues database programmers face when deciding how to design tables in a database.

Trains and Tables

I love trains, and often take the train between Tel Aviv and Haifa when I must travel between those two cities. One day, after leafing through the small paper schedule that Israel's Rail Authority distributes, I realized the implementation of a computerized train schedule is not as obvious as it would appear at first.

Rail schedules often come in the form of a printed table, with the columns representing individual trains and each station in its own row. Each table lists trains on one rail line, in a single direction.

Since relational databases store all of their data in tables, you might think this is a perfect way in which to store the information. In order to allow us to add and delete trains more easily, we will swap the axes from the printed schedule, putting the individual trains in the rows and the stations in the columns.

To define such a table in SQL, we could use a query like this:

```
CREATE TABLE HaifaToTelAviv (
   haifa_central   TIME NOT NULL,
   haifa_bat_galim TIME NOT NULL,
   binyamina       TIME NOT NULL,
   hof_hacarmel    TIME NOT NULL,
   ta_central      TIME NOT NULL,
   ta_hashalom     TIME NOT NULL
);
```

Given such a table, we could enter our trains as follows:

```
INSERT INTO HaifaToTelAviv
   (haifa_central, haifa_bat_galim, binyamina, hof_hacarmel,
     ta_central, ta_hashalom)
VALUES
   ("12:05", "12:10", "12:17", "12:37", "13:16",
    "13:21");
```

If you have any experience with databases, you can quickly see the terrible problems in store for us here. For starters, what happens if a new station is built between Haifa and Tel Aviv? That would require us to redefine our table, adding a new column, and that's only the beginning. It is a bit absurd that each train line requires two tables, one for each direction. And there isn't any way for me to determine whether a particular rail line serves any two cities—if the cities are represented by columns. What can we do about Tel Aviv? If two cities are close to each other and I can take a train to either one, I will have to query two tables in order to find the answer.

In addition, trying to query information from the above HaifaToTelAviv table would be difficult, requiring us to know the name of the column corresponding to each station. The problems just continue from there—for instance, what do we enter if the express train passes Binyamina? We could define the "binyamina" column to be NULL and enter a NULL value in that column. However, NULL normally indicates that a value is unknown or missing, whereas the reason in this case is much simpler.

Finally, what happens if a new schedule comes out, making each train later by a different amount of time? Editing the schedule in this format would be quite difficult.

Designing for Flexibility

How should we model the train schedule, then, if we cannot do so from the printed schedule? The solution is to break the information into smaller tables,

bringing them together to answer questions. Relational databases specialize in this sort of operation, allowing us to "join" two or more tables together.

Breaking the single large table into many smaller tables makes the database more flexible, allowing us to ask many more questions than would otherwise be possible. For example, we should be able to ask questions like:

- What is the last train from Haifa that will arrive in Tel Aviv before 11:00 a.m.?
- Are there any express trains from Tel Aviv to Haifa?
- What time will the 10:00 a.m. train from Binyamina get to Tel Aviv?

If we model our data correctly, breaking it down into sufficiently small and flexible tables, it should be possible to answer any of these questions with a single SQL query.

These examples all use MySQL, a "mostly free" database popular with many web sites. MySQL lacks some of the advanced features of other databases, such as transactions and referential integrity. However, it is easy to install and administer and is extremely fast. You can learn more about MySQL at http://www.mysql.com/.

For example, here is a definition of the RailStations table:

```
CREATE TABLE RailStations (
   id TINYINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   UNIQUE(name)
);
```

The only reason for RailStations to exist is to associate a numeric ID with each station. It might seem silly to create such a table, when we could enter station names directly wherever we need them.

However, giving each station an ID number gives us two advantages. First of all, we can be sure the station names will be spelled consistently, without variations in spelling, capitalization and abbreviations. Second, an integer consumes less space than the name to which it points. Each **tinyint** consumes a single byte, whereas a 20-character station name will consume 20 bytes. Referring to the full name would thus consume 20 times as much RAM and disk space.

Notice that we define **id** to be a column of type **TINYINT UNSIGNED**. This allows us to assign values between 0 and 255. Large rail systems, with more than 255 stations, would need to use a **SMALLINT UNSIGNED**, which ranges between 0 and 65535.

We ensure each station name in RailStations is unique by giving it the **UNIQUE** qualifier. The ID numbers are already guaranteed to be unique because they have been declared the primary key. Better yet, because we specified **AUTO_INCREMENT**, MySQL will automatically assign an ID number if an INSERT query ignores it. For example:

```
INSERT INTO RailStations (name)
VALUES ("Nahariya");
```

If we now query the database:

```
SELECT id
FROM RailStations
WHERE name = "Nahariya";
```

we learn that Nahariya has been automatically assigned an ID of 1.

We can insert one or more new rows into the table with a single INSERT statement. For example, the following adds several more rows to RailStations:

```
INSERT INTO RailStations (name)
VALUES ("Akko"),
       ("Hof Hacarmel"),
       ("Tel Aviv Central"),
       ("Tel Aviv Hashalom"),
       ("Lod"),
       ("Rehovot"),
       ("Herzliya")
;
```

Handling Rail Lines

Unlike cars, buses and airplanes, trains run along fixed lines. Each line must have at least two stations, and each station is on one or more lines.

We could have included an additional “lines” column in RailStations, identifying the line with which each train is associated. But given such a table, how would we handle stations that sit on more than one line? It would not make sense to treat the station as two separate stations, particularly if people will need to switch trains.

A better solution uses a separate RailLines table, defined similarly to RailStations:

```
CREATE TABLE RailLines (
   id TINYINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   UNIQUE(name)
);
```

Now that we have a list of lines and stations, we will create a third table that describes the intersection between the two:

```
CREATE TABLE StationLines (
   station_id TINYINT UNSIGNED NOT NULL,
   line_id    TINYINT UNSIGNED NOT NULL,
   north_to_south TINYINT UNSIGNED NOT NULL,
   UNIQUE(station_id, line_id),
   INDEX(station_id),
   INDEX(line_id),
   INDEX(north_to_south)
);
```

StationLines is a table that brings together the train stations and the lines on which they sit. **station_id** and **line_id** contain values from the "id" columns in RailStations and RailLines, respectively. And **north_to_south** is an integer value that counts the number of stops between the beginning of the line and the named station. Thus, the northernmost station on a rail line would be assigned 1, the next station would be assigned 2, the station after that would be assigned 3, and so forth.

Because each station can sit on more than one rail line, and each rail line contains more than one station, we should not use the **UNIQUE** modifier with those columns. However, we do not want the combination of any station and line to appear in the table more than once. We enforce this by naming both **station_id** and **line_id** as arguments to **UNIQUE**. Either of those columns may appear multiple times, but the combination of any two values may appear only once.

For example, the following row places **station_id 1** as the northernmost station on **line_id 1**:

```
INSERT INTO StationLines
   (station_id, line_id, north_to_south)
VALUES
   (1, 1, 1);
```

The following indicates that **station_id 7** is 11 stops from the beginning of **line_id 4**:

```
INSERT INTO StationLines
   (station_id, line_id, north_to_south)
VALUES
   (7, 4, 11);
```

Joins

With StationLines defined, we can begin to ask the database basic questions. For instance, we can list the stations on line two:

```
SELECT station_id
FROM StationLines
WHERE line_id = 2
ORDER BY north_to_south;
```

This query produces the following result:

```
+------------+
| station_id |
+------------+
|          6 |
|          4 |
|          5 |
+------------+
```

While this answer is indeed correct, it is not very useful. After all, why should I have to remember various stations' ID numbers in order to use the system?

Fortunately, relational databases permit us to join two tables, allowing us to connect the station's ID number with its name. To avoid confusing columns from the two tables, we name each column using the **table.column** syntax, separating the two with a period. And to reduce the amount of typing we must do, we give each table a nickname.

For example, we can structure our query such that it selects information from both RailStations and StationLines:

```
SELECT S.name
FROM RailStations S, StationLines L
WHERE L.line_id = 2
  AND S.id = L.station_id
ORDER BY north_to_south;
```

The query now produces the following results:

```
+-------------------+
| name              |
+-------------------+
| Lod               |
| Tel Aviv Central  |
| Tel Aviv Hashalom |
+-------------------+
```

Beginning database programmers often make the mistake of not qualifying their joins enough; that is, not putting enough statements in the **WHERE** clause. This is because a database server produces a join by combining every row in RailStations with every row in StationLines. The **WHERE** clause tells the server which rows to remove from the resulting table.

In the example above, the database server first creates a table of 112 rows (8 rows in RailStations x 14 rows in StationLines). It then removes all rows in which **L.line_id** is not 2, producing 24 rows. It then applies the final criterion, throwing out those rows in which **S.id** and **L.station_id** are unequal. The result is three rows.

Because we have broken down the data into three tables and we can join any combination of tables with any set of criteria, our database can already help us answer some basic questions. For example, which rail lines connect to the Tel Aviv Central station? Knowing the ID of that station is 4, I can compose the following query:

```
SELECT L.name
FROM  RailLines L, StationLines SL
WHERE SL.station_id = 4
  AND L.id = SL.line_id;
```

That query produces the following results:

```
+------------------------------+
| name                         |
+------------------------------+
| Nahariya  - Tel Aviv         |
| Tel Aviv  - Be'er Sheva      |
| Binyamina - Tel Aviv suburban |
+------------------------------+
```

If I join a third table in my query, I can use the station's name, rather than its ID number:

```
SELECT L.name
FROM RailLines L, StationLines SL, RailStations S
WHERE L.id = SL.line_id
  AND SL.station_id = S.id
  AND S.name = "Tel Aviv Central";
```

You might think the latter query, in which we name the station explicitly, would be the more common and useful when designing database applications for the Web. In fact, it's not. <select> lists and other HTML form elements distinguish between the value passed to the server and the value displayed to the user. For example:

```
<select name="station">
  <option value="4">Tel Aviv Central
</select>
```

The above one-element <select> list gives us the best of both worlds—it displays the station name to the user, but actually passes the ID associated with that station. This means our query can join two tables rather than three, which reduces the amount of memory it uses, as well as the speed with which results are returned to the client.

Indexing

In addition to the column definitions and the **UNIQUE** qualifier, our definition for the StationLines table included three **INDEX** lines—one for each of the **station_id**, **line_id** and **north_to_south** columns.

While it often helps to think of a relational database table as a glorified spreadsheet with rows and columns, there are some important differences. One is that a database table does not store its rows in any particular order. If we are interested in retrieving rows from the table in a certain order, we must specify it with the **ORDER BY** clause in our query.

Because rows are not ordered in any particular way, a **SELECT** query can often take quite a while to fulfill. For example, take the following query:

```
SELECT id
FROM RailStations
WHERE name = "Tel Aviv Central";
```

This might not seem like a time-consuming query, given that it involves a single table and a simple **WHERE** clause. But since the rows of RailStations are not stored in any particular order, finding the rows where the name is "Tel Aviv Central" can take quite a while. This might be a negligible amount of time in the case of a 100-row table, but when a table contains 1,000 or 10,000 rows, the time can become noticeable. In this particular example, the database server is probably smart enough to realize that RailStations.name has been declared **UNIQUE**, meaning our query will return one row, if it returns anything. This means the server will, on average, have to search through only half of the rows —but that can still take quite a while.

An index changes this picture by adding a pointer to each column value. If RailStations.name is indexed, the MySQL server can almost immediately find those rows containing a particular value. It can also determine whether a value exists at all.

If indexes can increase query speeds so dramatically, why are rows unindexed by default? The main answer is that indexes are written and updated each time an **INSERT** or **UPDATE** operation is performed on a table. Since the majority of database queries are **SELECT**s, in which the index can substantially improve performance, this is normally an acceptable trade-off. However, certain applications must **INSERT** and **UPDATE** at maximum speed, in which case creating an index can cause problems.

Since indexes are used in locating columns of a certain value, they are necessary for only those columns that will be named in **WHERE** clauses. There is no need to index a column that is displayed, but rarely used as a search criterion.

In some cases, it is enough to index the first part of each column rather than the entire column. For example, if we are indexing a column of type **VARCHAR(50)**, then we might be able to index only 10 of those characters. This will retain most of the advantages of a full index (since the first ten characters are rarely identical in such a text field), while reducing the amount of information the index must store.

Entering Train Information

Now that we have thoroughly examined the tables describing the train system, it is time to put some trains on those tracks. The question of how to model this data is a tough one, since there are a number of ways in which to accomplish it. I decided to split this information into two tables, Trains and DepartureTimes.

Each row of Trains describes a particular train, indicating the line on which it runs, the ID numbers of its origin and destination stations, and the time it departs from its origin:

```
CREATE TABLE Trains (
  id SMALLINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  line_id TINYINT UNSIGNED NOT NULL,
  origin_id TINYINT UNSIGNED NOT NULL,
  destination_id TINYINT UNSIGNED NOT NULL,
  depart_origin_time TIME NOT NULL,
  UNIQUE(line_id, origin_id, destination_id,
    depart_origin_time),
  INDEX(line_id),
  INDEX(origin_id),
  INDEX(destination_id),
  INDEX(depart_origin_time)
);
```

The first column is a primary key, allowing us to describe each train with a single number. The combination of a rail line, origin, destination and hour should be unique, so we ask the database server to enforce this condition with the **UNIQUE** keyword.

Finally, we define the DepartureTimes table, which stores information on when a train will leave from a particular station:

```
CREATE TABLE DepartureTimes (
   train_id SMALLINT UNSIGNED NOT NULL,
   station_id TINYINT UNSIGNED NOT NULL,
   departure_time TIME NOT NULL,
   INDEX(train_id),
   INDEX(station_id),
   INDEX(departure_time)
);
```

Once we enter information into these tables, we can start to perform sophisticated queries. For example, which trains arrive at “Tel Aviv Central” before 8 a.m.?

```
SELECT train_id
FROM DepartureTimes
WHERE departure_time < "08:00"
  AND station_id = 4;
```

Sure enough, this query returns a table containing two rows:

```
+----------+
| train_id |
+----------+
|        1 |
|        2 |
+----------+
```

Now we know two trains will arrive in Tel Aviv early enough for us to catch a morning meeting. But which trains are those? It would be nice to get more information than that. One possibility is to print the name of the origin station and the hour at which the train leaves:

```
SELECT S.name, T.depart_origin_time
FROM DepartureTimes DT, Trains T, RailStations S
```

```
  WHERE DT.departure_time < "08:00"
    AND DT.station_id = 4
    AND DT.train_id = T.id
    AND S.id = T.origin_id;
```

Notice how SQL allows us to use < and > when handling dates and times, for columns declared as **DATE**, **TIME** or **DATETIME**. Given the contortions one must use in order to compare dates and times in nearly any programming language, this built-in date comparison is still one of my favorites.

Assuming we want to take the first train of the day (**ID 1**), we can print the list of when it will arrive at each station:

```
SELECT T.id, S.name, DT.departure_time
FROM RailStations S, DepartureTimes DT, Trains T,
  StationLines SL
WHERE T.id = DT.train_id
  AND T.id = 1 AND T.line_id = SL.line_id
  AND SL.station_id = DT.station_id
  AND DT.station_id = S.id
ORDER BY T.id, SL.north_to_south
;
```

We can even print a full schedule for trains to Tel Aviv (**ID 5**):

```
SELECT T.id, S.name, DT.departure_time
FROM RailStations S, DepartureTimes DT, Trains T,
  StationLines SL
WHERE T.id = DT.train_id
  AND T.line_id = SL.line_id
  AND SL.station_id = DT.station_id
  AND DT.station_id = S.id
  AND T.destination_id = 5
ORDER BY T.id, SL.north_to_south
;
```

Finally, we can retrieve a full schedule for trains to Tel Aviv (**ID 5**) that leave after 9 AM:

```
SELECT T.id, S.name, DT.departure_time
FROM RailStations S, DepartureTimes DT, Trains T,
  StationLines SL
WHERE T.id = DT.train_id
  AND T.line_id = SL.line_id
  AND SL.station_id = DT.station_id
  AND DT.station_id = S.id
  AND T.destination_id = 5
  AND T.depart_origin_time > "09:00"
ORDER BY T.id, SL.north_to_south
;
```

Conclusion

While it is often a good idea to use a database for storing and retrieving information in a web application, it is not always obvious how to go about structuring the tables in that database. Splitting information into separate tables, as we have seen, makes it possible to mix and match data in a wide variety of ways. By using numeric primary keys and indexing the columns we will need most, we can make our queries efficient as well as flexible.

Now that we have seen how to define our database tables in an intelligent way, it is time to create some applications to use them. Next month, we will look at a variety of applications that can use these tables, giving them interfaces appropriate for web users.

Reuven M. Lerner , an Internet and Web consultant, recently moved to Modi'in, Israel following his marriage to Shira Friedman-Lerner. His book Core Perl will be published by Prentice-Hall in the spring. 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

Commodore 64 Game Emulation

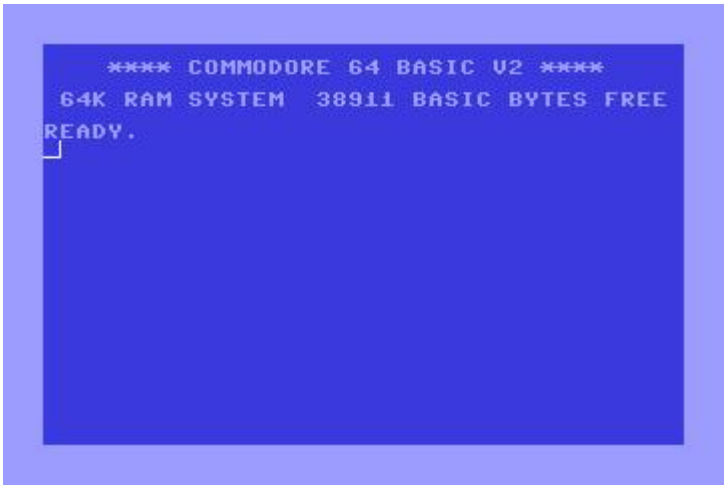
Jason Kroll

Issue #72, April 2000

There must have been some magic in that old grey box...

This month, “Games Focus” moves out of upFRONT and stands on its own as “Games We Play”. Last month, we went a little nuts, or rather I went a little nuts, over emulation of the arcade games we used to play in pizza parlors, roller rinks and video arcade joints. Those games were really fun, and honestly, every time I look at the new ones coming out, I think to myself how happy I am not to be in the video-game industry, because the coding involved in the crazy 3-D things done in these games is too complicated. How *do* you figure out which parts of an object are sitting behind the others, and how the light and shadows fall? Those old games were really neat, and some day, when you've got a gigantic flat-screen monitor, you can have your own at-home video arcade. Sit on a sofa, drink iced tea and fire up the old video games on your wall (of course, we'll have wireless controllers, too). Ah, the good old days, er...

As there is another Skywalker, there is also another area of emulated games. The Commodore 64, as many of you will remember fondly, is an 8-bit computer based on the 1MHz 6502 processor, which is now emulated quite well on Linux. It was wildly popular and fun to hack. It came with BASIC, but you could get a cheap assembler anywhere and start exploring exactly which bits did what. In fact, you could just use the BASIC command **POKE** to upset bits all over the place, often leading to disastrous effects. Recite numbers like 64738 or 53281 to some old C64 user, and you're sure to get a rise. Unlike Linux, the C64 was a single-user machine that was effectively in root all the time and functioned very well for real-time demands such as video games. And that brings us to the point.



Commodore 64 games are now available for you to play again on your Linux box. They are exciting, with a feeling unlike any other computer games. Unlike the emulated video arcades, where you can play roughly 2,000 games, the C64 emulators open the door to over 10,000 games as well as countless demos (if you don't know what these are, you should have a look) and other odds and ends. Of course, these games were a bit odd (programmers had more free reign to express themselves in home computer games), often with lower-quality graphics than the coin-ops and much more interesting and creative game play. Remember, many of the best games simply aren't suited for coin-op arcades because they're slightly long term. Indeed, computer games were once for those with longer-than-a-quarter attention spans. One thing I find particularly amusing is to go back to those old chess programs that infuriated my eight-year-old self, and beat on them mercilessly (they were *scary*; Sargon even came on a sinister red disk). Well, chess is one long-term game, but RPGs (role playing games) go on far longer. Anyone for Bard's Tale or Ultima? The AD&D (Advanced Dungeons and Dragons) series from TSR Strategic Simulations (a division of Interplay)?

There was nothing new for Commodore 64 gamers when games like Mortal Kombat hit the streets. Years before, C64 folks had played the original combat games, such as Melbourne House's Kung Fu (Way of the Exploding Fist) or Epyx's International Karate+. More creative martial-arts games were available, including classics such as Bruce Lee, Yi Ar Kung Fu (with the protagonist Oolong) and countless others. It was also no surprise when, years later, Sid Meier rose to stardom in the PC world on his Civilization series; Sid was already a longtime hit with C64 fanatics for ingenious masterpieces such as Pirates! And, it was nothing new for the C64 crowd when Lord British went on to celebrity status with his Ultima series (in its current incarnation as Ultima Online). These chaps have been around forever, so it has been a nostalgic surprise to see them featured recently in magazines—video game designers, fancy that!

Current hackers, especially hacker historians (if such animals exist), techno-cultural anthropologists and other similarly interested folks have another value for this software; that is, the historical-cultural value. Linux hackers often claim lineage dating back to UNIX and whatnot, but the fact is, we can't all have come from UNIX. Someone must have been from that generation to whom BASIC was a language and for whom computers were single-tasking bit boxes. Indeed, the excitement of the early personal computer revolution had little direct involvement in UNIX; it was the Apples, Commodores, and to some extent, the IBM PCs (while some may even remember the Spectrums). The C64, for whatever reason, attracted more ingenuity and technical exploration than the others, and developed its own extremely vibrant and enthusiastic scene (which later migrated to Amiga). Even with software that didn't cost very much and fewer users than the computer world has today, Commodore somehow inspired seemingly limitless creative energy and development.

Commodore 64 software relates to a time before the Internet, before hard drives, before open source, before free software was popular, before compilers, before networking. Although these things existed, they didn't affect the C64, since we didn't have the resources for anything more than short machine-code programs. The culture was much different then, with software houses pumping out games high in playability and imagination, low on resources and abundant in programmers able to express themselves in their wares. (The computer community was much brighter and more creative than today's mainstream Windows world, so it was also more receptive to creativity.) After watching many people make a fortune on commercial software, groups of teenagers sprung up around the world, oriented around trafficking unauthorized copies of software (struggling to be the first to break the copy-restriction schemes, which were actually very harmful on the disk drives), writing intros and demos, maintaining contacts and hosting enormous demo competitions and copy parties. Demo writing, in particular, often overshadowed the cracking activities. Some relics of this culture survive even today, but the Net has largely homogenized the modern computer world. Those curious about the cultural history of '80s hackers (when hackers broke into computers, crackers broke software restrictions, and coders wrote code) could find much of interest in C64 wares, especially the demos, intros and “cracktros”.

Demos would demonstrate what the machine was capable of and what the coder could contrive to do. Typically, demos contained scroll texts full of greetings (which you may remember from early episodes of “Stupid Programming Tricks”), star fields (essentially little white pixels that moved), music (known as SIDs on the C64) and some graphics or other animation. What

was outstanding, of course, from a technical standpoint was how far these cats could push the computers. Just try sitting down at your Linux box and writing a demo with a sine scroller (text that warps like a slithering snake), music, star fields, animation and a background, and you'll probably take up more than 1MHz of your system's resources. Now try doing that in 6502 assembly code in the days when there were no easy graphics libraries and no one shared source code. (Because, if you knew how to do what other people could do, you'd be just as cool as them; hence, they never gave away their secrets.) In fact, the horrible secrecy surrounding source code sharing meant everyone had to figure out *everything* for themselves, which wasted a lot of time. (Admittedly, this problem was much more severe in the states; European sceners were usually better about sharing.) Often, people would “trade” programming tricks the way many traded cracked software; you'd have to give the code to something cool, or they wouldn't help you out. It seems quite selfish; well, that's the culture of the '80s. At least I can say *I* have always shared code...

Intros and cracktros were small demos which often didn't do anything enormous or particularly outstanding outside of the basic music and scroll text, with some small additions, usually for the purpose of saying “So-and-so cracked this game, we're so cool, megagreets to so-and-so, such-and-such, who-and-who, et al.” and maybe with some insults to their schoolteachers or bus drivers (strangely, for really smart kids, school is usually a nightmare). Intros are quite interesting for the “8-bit (unsigned) IQ gang graffiti” cultural value. They come attached to “cracked” games, although now they've been catalogued separately for those wanting the intros and not the games.

Games, however, were nominally the point of all this. Fortunately for the lawful crowd, the companies that made these old C64 games are largely defunct, so the copyrights don't appear to have any owner (not that everyone considers copyright law to be legitimate). In fact, in the few cases where old programmers have found their games on-line, they have uniformly been delighted to see people playing their old games. Unlike the MAME (multiple arcade machine emulator) project where some industry groups put up a fuss (we assume because they're afraid of emulation spreading to modern console games), there hasn't been a fuss over old C64 wares. Many of us actually paid for the software years ago, so we're probably even legally protected, although there hasn't been much of a problem. One group, the IDSA (Interactive Digital Software Association), tried to shut down some C64 sites, but since they're just a trade group (like the MPAA [Motion Picture Association of America] or RIAA [Recording Industry Association of America], our villains of the moment) they haven't been very effective, as they don't actually represent the now-defunct companies who once held the copyrights. So, be a tad careful just in case, even though I've never heard of anyone getting in trouble over it. (If you're morally opposed to legal strong-arm tactics of threatening web sites with fleets of

lawyers, maybe you'll want to collect and spread old C64 games just to be oppositional to the IDSA.) However, if you want to be perfectly pure, you can just stick to the demos and intros, which are very interesting and inspirational.

The Commodore 64 emulators for Linux are known as ALEC and VICE. ALEC is actually my favorite because it runs in console (I live in console), whereas VICE is for the X Window System. Nevertheless, VICE is much more full-featured and will run many programs that ALEC will not. Also, it's released under the GPL. They're both small programs and I recommend getting both of them, particularly because you'll need the ROM images that come with VICE. It's obvious and easy to get started; you can even download binaries if you can't deal with compiling. These projects haven't been maintained in a while, since they're basically complete and don't need much further work, so I don't know of any active web pages. Hence, grab them via FTP from ftp://metalab.unc.edu/pub/Linux/system/emulators/commodore/ (or http to the same address, if you prefer).

As for where to find C64 software, I recommend scouring the Web as you would a used-book store, treasuring rare finds and the web pages of old demo groups. You *could* go straight to a 10,000-game archive, but that's no fun. Try entering “Commodore 64” or something similar into a search engine—that's fun! If you have to start somewhere, the obvious http://www.c64.org/ and .com are sure to link you to the rest of the Commodore scene. The master collection of C64 scene member interviews known as *In Medias Res* resides at http://www.imr.c64.org/ and is intriguing for those who want to know what coders/crackers of old were truly like and where they are today. In particular, you can read how the old scene struggled with internal divisions and questions of friendship, community, principle, newbies, lamers, the law, money and the other problems that now confront Linux. If you just want to listen to Commodore 64 music on your Linux box, get Sid Play and check out the High Voltage Sid Collection at http://home.freeuk.net/wazzaw/HVSC/, but don't forget the games!

Jason Kroll (info@linuxjournal.com) is technical editor of *Linux Journal*. He wonders how history will recall the creation and popularization of free source philosophy, open-source methodology and the success of GNU/Linux.

Focus on Software

David A. Bandel

Issue #72, April 2000

webnotebook, darxite, AoNettool and more.

The RPM standard is a panacea for those who use RPM-based distributions (Red Hat, Caldera, Mandrake, SuSE, etc.). Or is it? You may have noticed that RPMs from one distribution don't always work on another. They don't install due to dependency problems (even though they would run just fine), or they install but segfault (library incompatibilities), etc. So you grab a .src.rpm file, and try to build it. Often, that doesn't work because some distributions have seen fit to make slight alterations to their version of RPM. What about those who can't even build an RPM .spec file and the patches required? Well, you might want to check out http://specs.pananix.net/. The .spec and patch files are thin at the moment, but with help from the RPM community, this could be a great site for grabbing a .spec file and patches that work for your distribution. Hopefully, all that's required is the software archive file, the .spec and patches. The command

```
rpm -bb && rpm -i ../RPMS/i386/
```

should do the trick.

webnotebook: http://www.entropia.com.mx/~roadmr/webnotebook/

This is a small web notebook with fields for subject and body. It automatically adds the creation date, modification date and a note number. **webnotebook** is similar to **note**, but is used via a web browser (lynx works just fine). You can search for any term, and the notes found will be listed. Searching on nothing returns everything. Unlike note, which sorts on note number, the default sort in webnotebook is the modification date (latest first). It requires MySQL server, web server configured for CGI, Perl and the Perl DBI/DBD module.

darxite: http://darxite.cjb.net/

Much like Downloader for X, this program will download a number of files. While slightly more complex than Downloader for X, **darxite** runs a daemon in the background, which you can connect to in order to pass URLs for downloading (including downloading directories recursively). Several client programs, for X and the command line, can be used to pass commands, URLs and monitor download progress. This program will continue downloading even after you log out. It requires libpthread and glibc.

AoNettool: 134.130.48.9/~alex/AoNettools-0.9a-rev1.tar.gz

This tool is a Tcl/Tk front end for **ping**, **traceroute**, **nslookup**, **finger** and **whois**. Output is displayed in the window and can be saved for future reference. This tool works well for those not accustomed to a command line. I would only add **netstat -rn**, **netstat -an** and **ifconfig** to have a relatively complete set of tools for viewing the network status of the host. It requires Tcl/Tk.

Advanced Packet Sniffer: http://www.swrtec.de/

This packet sniffer provides a wealth of information about packets on your network, as well as a display of the entire packet. The header is put into human-readable form and presented, starting with the MAC address. A number of options are available, such as whether to include or exclude a particular MAC or IP address. This sniffer can be run only by root, but that constitutes a fairly good reason not to allow rogue Linux systems on your network. It requires glibc.

filetraq: http://filetraq.xidus.net/

This particular utility seeks to replace the no-longer-free **tripwire**, but **filetraq** works from a slightly different perspective. Rather than track large numbers of files daily, it looks for changes in a few key files at very short intervals. Changes in any of these files are reported via e-mail when the change is detected. Key files include one of /etc/passwd or /etc/shadow and a few others, such as /etc/services and /etc/inetd.conf and easily editable startup files. Basically, filetraq uses **diff** to compare the file against the database, so you could include /bin/login also. It requires bash and recommends **cron**.

anteater: http://www.profzone.ch/anteater/

This utility will search through your mail log file using **grep** and display important statistics gleaned from that file: largest messages, source of most messages, etc. One problem this particular utility suffers from is the lack of a usage message. Both **-h** and help tell you to read the manual, but the manual

isn't installed with the software—an oversight, I'm sure. It requires libstdc++, libm and glibc.

nutcrack: http://www.birdnest.org/zzrhear/

It has been awhile since a new password-cracking program has appeared. The ones currently available are difficult to compile or require mega-swap space and more resources than most modest systems have. **nutcrack** is quick. One caution: the dictionary used is extremely important. I created a new account, and used the account name as the password—it wasn't caught. So, be sure to add the contents of your password file (user name and GECOS field) to the dictionary. It requires Perl and a good dictionary file.

weedlog: http://www.firepool.com/weedlog/

This particular daemon is yet another IP logger, but with one big difference: unlike many loggers, this one does not rely on promiscuous mode and logs the packets in an easily understood, human-readable format (configurable). This makes for fewer mistakes when manually reading the log, and it can be machine-processed easily. You can choose among logging TCP, UDP, ICMP or IGMP (or any combination). Caution: if you get many simultaneous connections, you will see a **weedlog** spawned to report each one (it will die, but you could end up with a very large process table). It requires libpthread and glibc.

David A. Bandel (dbandel@pananix.com) is a Linux/UNIX consultant currently living in the Republic of Panama. He is co-author of Que Special Edition: Using Caldera OpenLinux.

Archive Index Issue Table of Contents

Advanced search

Embedded Linux News Briefs

Rick Lehrbaum

Issue #72, April 2000

The latest on Lineo, Transmeta, LinuxDevices and more.

Lineo unveiled a challenge to Microsoft's Windows CE: Embedix PDA. The product will add a Win CE compatibility layer to Linux, allowing developers to easily port Win CE applications to Linux. Lineo's target is palm computers and embedded systems. (http://www.lineo.com/)

Transmeta disclosed plans for "Mobile Linux", which will support the highly constrained resources of portable Internet devices and embedded systems. The company plans to offer its Linux enhancements to the Open Source community. (http://www.transmeta.com/)

LinuxDevices.com announced the results of its third Embedded Linux Poll, which asked developers to describe an embedded computing application in which they're planning to use Linux. The poll's results can be viewed at http://www.linuxdevices.com/polls/.

A new white paper from **MontaVista Software** reviews the benefits of Linux to embedded applications, discusses the alternatives available, and offers a glimpse of what to expect from embedded Linux in the coming year. (http://www.mvista.com/)

Corel Corp. entered into an agreement to acquire up to 30% of start-up OE/ONE.com, a company founded by a former Corel executive that has developed a sub-$500 Linux-based Internet appliance. (http://www.corel.com/)

Lineo announced it has begun shipping Embedix Linux 1.0, the company's embedded Linux distribution. Embedix is targeted at x86 and PowerPC-based embedded devices. It requires a minimum of 8MB RAM and 3MB of ROM/Flash memory and is based on the Linux 2.2 kernel. (http://www.lineo.com/)

Evidencing significant inroads made by Linux within the U.S. government, the **National Security Agency** (NSA) awarded a contract to Secure Computing Corp. to develop a robust, highly secure configuration of Linux.

Lineo, Inc. announced a major embedded Linux design win in the set-top box market. The system, to be marketed by Bast, Inc., will go in hotel rooms and apartment buildings. The initial plan is for 50,000 systems, priced at $285. (http://www.lineo.com/)

Touch Dynamics announced an open-source project to develop KOSIX, an industry-standard public kiosk terminal operating system based on Linux. KOSIX will offer an open-source alternative to conventional, proprietary kiosk OSes. (http://www.touchdynamics.com/)

Red Hat announced appointment of Michael Tiemann as the company's new Chief Technical Officer (CTO). Tiemann, co-founder of the recently acquired Cygnus Solutions, is the principal architect of the EL/IX embedded Linux standard. (http://www.redhat.com/)

An interview with Michael Tiemann, **Red Hat** Chief Technology Officer, discusses the impact of Red Hat's acquisition of Cygnus on the embedded Linux market and the future of the Cygnus EL/IX Embedded Linux API initiative. (http://www.linuxdevices.com/articles/)

IBM disbanded its Internet division and redirected its resources toward an aggressive campaign to promote Linux. As part of this strategy shift, former Internet division executive Irving Wladawsky-Berger was transferred to the new IBM Linux group. IBM says it will collaborate with the Linux open-source community and has dedicated a portion of its web site to Linux-related information. (http://www.ibm.com/linux/)

DataViews Corp., a provider of “human machine interfaces” (HMIs) for factory automation operator interfaces, announced a Linux version of its high-end HMI software tool, DataViews. The company claims to be the first provider of Linux-based HMI tools.

The organizational meeting of a new **Embedded Linux Consortium** will be held at the Embedded Systems Conference in Chicago on March 1. The non-profit group plans to serve as a trade association for companies in the embedded and real-time Linux market. (http://www.linuxdevices.com/forum/)

UC Berkeley announced two short courses on real-world applications programming, to be offered this spring in Los Angeles, San Francisco and Boston. The courses will emphasize Linux and are entitled “Real-Time

Programming for Embedded Systems" and "32-Bit Real-Time Operating Systems with an Emphasis on Linux". (http://www.berkeley.edu/unex/eng/)

There's a movement afoot to develop Linux-based "programmable logic controller" (PLC) technology. PLCs are commonly used in manufacturing and factory automation control systems. A Linux-PLC web site, mailing list and open-source software are being created. (http://www.linuxplc.org/)

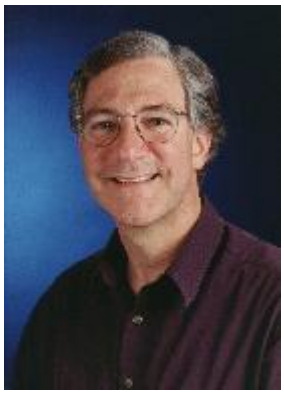
FSMLabs released a beta version 3.0 of RTLinux for "hard real-time" applications. RTLinux can control machinery while maintaining full Linux compatibility. The new release, based on the latest Linux 2.3 kernel, offers improved performance and supports ports to non-x86 architectures. (http://www.fsmlabs.com/)

Linux received a boost in laboratory and industrial test, measurement and control with the announcement of comprehensive Linux support by **National Instruments**. The company has assembled Linux-based instrumentation and control solutions for VME and VXI-based hardware. (http://www.ni.com/)

Intel began delivering prototypes of Itanium, its new 64-bit CPU (formerly code named Merced). Sources within Intel said the company will shortly authorize the Trillian group (a team working on Linux for Itanium) to release the Itanium Linux source code to the Linux developer community.

MontaVista Software Inc. released its "Hard Hat Net" CompactPCI backplane networking package to the GPL open-source community. The move provides developers with powerful networking options for using Linux and CompactPCI in telecom, telephony, Internet and other embedded applications. (http://www.mvista.com/)

VA Linux Systems introduced **SourceForge**, a major open-source initiative that provides over 700 open-source development projects with extensive hosting and communication resources. The services are available at no cost to open-source developers. (http://sourceforge.net/)



Rick Lehrbaum (rick@linuxdevices.com) co-founded Ampro Computers, Inc. in 1983. In 1992, Rick formed the PC/104 Consortium and served as its chairman through January 2000. In October 1999, Rick turned his attention to embedded software, founding LinuxDevices.com—"the Embedded Linux Portal".

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

e-Market e-Madness, e-Nough.

Stan Kelly-Bootle

Issue #72, April 2000

Of course, science, life and society being what they are (send me a $1,000 check for this month's definitions), the hard/soft debate is doomed to waffle on.

There are, they say, hard and soft sciences, not to be confused with difficult vs. easy, but more related to claims of objective, measurable, repeatable precision (we stout hardies) versus subjective, hand-waving, proofs-by-assertion (you big softies). The spectrum of fuzziness typically ranges from pure mathematics and physics at the diamond-tipped top, then descends via chemistry and biology (almost converging to synonyms), ending with a series of “life” and “social” sciences. The latter at least co-opt the intentions and vocabulary of the scientific method and rightly escape the “beyond-the-soggy-pale” category of pseudo-science (astrology, UFOlogy, pyramidiotology, hidden-Bible cryptology ... *ad astra, ad nauseam*). Honest “theology” wriggles through the colander by rejecting the “scientific” model.

Of course, science, life and society being what they are (send me a $1,000 check for this month's definitions), the hard/soft debate is doomed to waffle on. The paradox is that comparing the hardnesses of any two disciplines requires a valid metric—and that metric will depend on the hardness of the hardest involved domain. You can hear the magic predicate *meta* creeping into the equation. It reminds us that the very foundations of the purest of pure mathematics (formal set theory) were softened (nay, Osterized and Cuisinarted) by Gödel's meta-mathematical shocks in 1931.

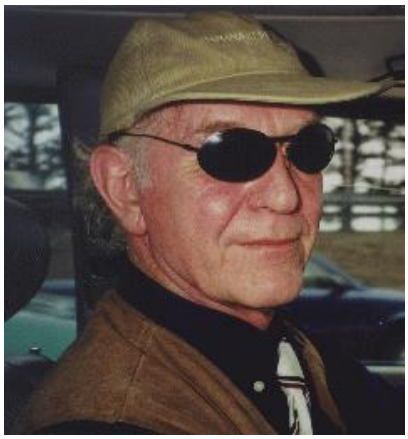
It's still difficult to accept that the hardest queen of the sciences proved so brittle after millennia of complacency. Harder still to note that everyday mathematics and all the dependent sciences rumbled on, regardless. However, there was a clear dent in the traditional hierarchy.

We may avoid an arithmetical **operator>>** with a sort of “diamond-scratches-glass” ordering, but that moves us to unprovable, contentious, domain-

dependent, metaphorical comparisons (e.g., in Political Science, we readily declare that Lincoln was a better president than Clinton).

There are other parameters available for comparing sciences. Some, such as “usefulness”, are possibly more determinable and worthy of discussion. Thus, cosmology (already disputed in the hard/soft science ratings since we can't, as yet or ever, repeat the “big bang” experiment, do a tachyon glide into the local wormhole, or contact our nearest parallel universe) seems to offer useless theories and predictions: whether the cosmos expands forever or collapses after 10 billion years is, to most readers of *Angela's Ashes*, hardly even worth mentioning.

What of our provably most useful sciences: economics and computer science? Demi-soft economics has been dubbed the “dismal” science, while semi-hard computer science must surely be the most boring (most of its formal results, dated 1935, are proofs that “this is impossible—don't bother!”) Yet hand-in-hand, both sciences have triggered the most undismal, unboring IPO stock-market episodes since the South Sea bubble. What is anything *really* worth? Forget the Marxian cosource-sweat-value-added axioms. If dazed bidders offer $1 million for a single Yahoo Japan share, then that is, by definition, its current “worth”. Next question? Red Hat may or may not fairly distribute its IPO gains, but they missed out 500% by not calling themselves e-Red e-Hat e-Linux. And just wait until I launch e-skb-gnu-e-free-hardware.com.



Stan Kelly-Bootle (skb@crl.com) has been computing on and off since his EDSAC I (Cambridge University, UK) days in the 1950s. He has commented on the unchanging DP scene in many columns (“More than the effin' Parthenon”--Meilir Page-Jones) and books, including The Computer Contradictionary (MIT Press) and *UNIX Complete* (Sybex).

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #72, April 2000

Readers sound off.

Correction

The OS of the People

I'm writing to express my entire support for what Jason Kroll has written in his article “Musings on Linux Profiteering” published on the *Linux Journal* web site (http://www.linuxjournal.com/articles/misc/012.html). I've always had the same fears and beliefs, especially because of the love I feel for GNU/Linux, the one and only OS that actually *came* to me, without asking me for money or anything —on the contrary, offering me great amounts of knowledge I wouldn't have with another OS. The moment I noticed the existence of GNU/Linux, a big change was made in the course of my life. In a country where you have to pay to get proper education, these things mean a lot to all of us—the people who don't have the resources to pay for private education. Not only to us, but to all people who in some way work with computers, providing a way to expand and improve our knowledge.

That's why I fear the monster of money so much, and the harm it could cause the entire GNU/Linux community. And this is why I'm not using FreeBSD, regardless of its good aspects: its license promotes commercial development, allowing anyone to prohibit the use of the software to someone like me, who does not have the resources to pay for software.

Thanks a lot for the article; it could help people realize how much they have to protect.

—Pablo Baena greyheart@fnmail.com

Love that Penguin

It was great to see Feathers McGraw featured on your magazine cover. I accidentally discovered Wallace and Gromit while flipping channels a few years ago. PBS was playing two of Nick Park's features; we later purchased a tape of “The Wrong Trousers”. These movies have become favorites of the entire family. The Linux box I set up at work is named Feathers after this character, because he's an outlaw. Hopefully that will soon change.

—Steve Brant steve@brant.nu

Upgrade Error

Kirk Petersen's “Using the Red Hat Package Manager” (January 2000) is a nice article, but has one problem. In the paragraph on “Upgrade Mode”, the example given:

```
rpm -u penguin-3.26.i386.rpm
```

will return the following error messages: “error: --u and -uninstall are deprecated and no longer work.” and “Error: Use -e or -erase instead.” At least, it will if you use RPM version 3.0.2 as I do. The command:

```
rpm -Uvh penguin-3.26.i386.rpm
```

would work much better. The man page is not very clear, but the info page calls it out correctly.

—Joe Luker spooky@cnmnetwork.com

How About that Date?

I noticed that even though the cover page of the *Linux Journal* had the right date (JANUARY 2000), all the inside pages on the lower right were stamped JANUARY 1900, with an editor's mark changing it to 2000.

Was this intentional or just another Y2K bug? Either way, it bothers me that I didn't notice it until the second reading.

—Arthur Hammerschmidt ajhammer@cgocable.net

It was intentional. A continuation of our joking claim to be Y2K-compliant on the December cover. The idea was to provide a laugh at all the Y2K uproar and ourselves —Editor

Y2K Strikes *LJ*

I usually do read my *LJ* before any dust collects on it, but I'm just now getting my fix with the January issue. In the “thanks, I needed that” kudos department, I really got a huge laugh out of your date. Thanks.

—Greg Edwards Greg.Edwards@usa.alcatel.com

In Linux We Trust

I would like to acknowledge Phil Hughes' article, “Linux Public Trust” (*LJ* web site, http://www.linuxjournal.com/articles/buzz/027.html). How timely it seems, given the announcement today of the Linux Open Source Expo & Conference Program to be held in Sydney, Australia in March. All the “big names” of the Linux world will be there, with keynote addresses by Robert Bishop of SGI and Robert Young of Red Hat, Inc. Bob Young's credentials include “a career in the computer finance arena, with 20 years of computer industry finance and marketing... which allowed him to see an opportunity with a phenomenon called Linux in 1993.” And here I was, thinking these guys were just hackers in suits!

By the way, the price of a ticket on this gravy train is $1350 AUS. For an extra $125, you can attend the TUXedo Night, featuring none other than John “maddog” Hall. How many other Linux luminaries are attending? Only time will tell. Perhaps one (or more) *Linux Journal* staff would also like to attend, if only to let the rest of us poorer Linux users know how the other half live. Public Trust? I'll drink to that, but with a virtual beer, at home in front of my computer.

Thanks for what is still an informative and enjoyable magazine, even if the suits appear to be more numerous these days.

—Laurie Darerough planet@ozemail.com.au

Spinning the Web

I read with interest Doc Searls' comments on operating systems being used on web servers (upFRONT, January 2000). My own site is being hosted by Virtualis, and they use FreeBSD quite extensively for virtual hosting. Their dedicated servers come with two options, namely NT/FreeBSD. Upon enquiring for a client, one of the support staff for Virtualis stated that NT is buggy as far as virtual hosting is concerned.

To come back to the Media Metrix survey, two of the five sites being hosted on NT belong to Microsoft—namely, msn.com and microsoft.com. I guess one can ignore those two, which brings the NT tally to 3...

Thanks for a most enjoyable magazine!

—Johan Pretorius JohanPretorius@flysaa.com

Ranting and Raving

Do you wanna know what really pisses me off about Linux? It's touted as a "free" OS (in other words, you can get it without paying a cent) but it's obviously *not*. I'd like to try Linux and see if it really *is* a good competitor to Windows. I understand there is *very little* software sold that can actually run under Linux, but I am still interested to see how the OS works. My question is simple: why pay some vendor my hard-earned money for an OS that finding software for is such a chore? And why should I trust anyone that says "it's free" and then wants to charge me for it?! I understand it costs money to make and distribute CD-ROMs, but why can't I simply download it if it really is free? Your web site *says* I can, yet I see no link to the FTP site. It doesn't take a brain surgeon to spot this lie.

As for Windows, I firmly believe Microsoft *earned* their market share by the fact that they created a terrific OS that a vast majority of computer users love. It's called supply and demand (duh). Yet I tend to scratch my head in wonder when I hear about some OS that's being touted as better (more stable), and free (when I can't even find the free software on the Internet) and *especially* when I don't see software that says it works under Linux. Usually, when a better product exists, there is a very competitive market for it (such as cars—Ford and GM are a perfect example).

My conclusion is this: Linux is nothing more than a cult following, like Apple's Macintosh. Mac users are cut from a strange cloth. They seem to insist on using a Mac, simply because they want to be different. They also appear to hate Microsoft, just because Microsoft has created some of the best software on Earth and *millions* of people gobbled it up like turkey sales at Thanksgiving.

As for the (corrupted) Justice Dept. vs. Microsoft... well, just because someone makes a superior product that has overwhelming customer demand doesn't make them a criminal enterprise.

—Scott Moore scottbomb@hotmail.com

Actually, we didn't think it took a brain surgeon to find Linux on the Internet. Just click on our "How to Get Linux" button, and you will find many download sites. More and more software products support Linux every day. Check out the ads in our magazine and the Linux Software Map at http://www.ExecPc.com/lsm/ —Editor

upFRONT

Various

Issue #72, April 2000

Stop the Presses, *LJ* Index and more.

Stupid Programming Tricks → Graphics glide across glorious ground

Welcome to another episode of typical brilliance. Yes, it's time for another round of Stupid Programming Tricks. We've been doing tricks for a while, and we can make some neat stuff, but are we as cool as the guys who wrote arcades such as Ms. PacMan, Arkanoid and Joust? Those were pretty fun video games, even if they were 2-D and took place on stationary backgrounds. The point is, software can be truly fun even if it's simple, and the most elegant game would likely be the simplest. (I imagine you can find Go enthusiasts who would agree.) Now, imagine if thousands of people across the world were playing a video game you wrote—that'd be pretty amazing.

The C language, libraries and GNU/Linux tools make it possible to develop software quickly, and game programming is one of the very best ways to learn how to code. It involves challenges of interface design, graphics and animation, real-time scheduling, collision detection, physics, sound, hardware interfaces (such as keyboard, joystick and mouse), as well as limitless creative scope. Best of all, game programming techniques are applicable to *any* kind of programming, from audio and video applications to user interfaces and indeed any program from “Hello World!” on up.

Programing an entire game is a mildly difficult project that could take a few days, but we'll look at one of the most basic elements, something applicable to any application involving graphics. We've already discussed how to initiate console graphics, how to scroll text across a screen and how to play music and sound. Now, we'll take a look at something simple but important: how to move an object smoothly over a background without damaging the background.

For our purposes, we'll start with the usual SVGAlib (you know, runs-as-root-not-very-safe SVGAlib), and initialize two screens: our virtual screen and our physical screen. The standard procedure, as you may remember from previous episodes, is to do all our drawing to the virtual screen (because it's *much* faster and flicker-free), then copy the virtual screen to the physical screen and hold it still for 1/60th of a second (the duration of a vertical refresh, which could even be 1/85th of a second on really fast equipment). For our background, we'll generate something cool by drawing a few lines and applying some sine equations to the pallette. As for the shape we intend to slide over the background, we'll use some equations to make a very funky square. Obviously, it's a small step to go from this situation to loading a custom-drawn background and an animated character, but we'll stick with our simple approach, largely because algorithms that generate graphics are inherently cool.

If you draw a background and then move your graphic across the screen, you'll leave a trail of pixels and ruin the background, which often happens periodically on slow computers running X. (You may remember the grey smears caused by moving a window, especially in Netscape and advanced window managers.) One solution is to redraw the entire background, then stamp your graphic in its new location. The problem with this approach is it severely wastes processor resources (you'll be redrawing the background at least 60 times every second), which makes for slow programs. Personally, I favor fast code and hack-job optimizations over structurally organized, procedural code; for example, loops instead of recursive algorithms. Redrawing the background often makes sense, if you're making changes to the entire background or if you do not have hardware scrolling capabilities. Suppose you have a 320x200x256 background (320 pixels wide, 200 pixels tall, with 256 8-bit colors), and you want to move only a 16x16 graphic across it. Think Ms. PacMan. Why not just draw the heroine and the ghosts? Why redraw the maze all the time, if it doesn't change? How much is actually moving?

The procedure we'll use is known as the cookie-cutter technique. It involves making a backup copy of the area onto which you want to stamp your graphic (on an Amiga, we'd call it “blitting” rather than “stamping”). Essentially, you make a backup copy of the target area, stamp your graphic over that area on the screen, hold it for a vertical refresh (so that the image is present long enough for the eye to register it), then replace the stamped area with the backup (cookie-cut) you made, stamping the background over your old image, and start again. As long as your graphic uses transparent or masked bits (instead of stamping solid black), you'll see your graphic sitting nicely on top of the background. It's the same situation as laying a graphic on the background of a web page, only here it's moving.

In our actual program, you'll notice we're using a physical screen and a virtual screen, when we could get by if we used only a physical screen (after all, copying a virtual screen to a physical screen is a more processor-intensive procedure than copying 16x16 pixels of background to a backup buffer, blitting a 16x16 image, and replacing the 16x16 background segment). The point is that drawing to a virtual screen is very fast, and if you ever want to do more than a single moving box, you'll be very happy you're already using the virtual screen. Extensibility is a good idea, especially when you leave things open for your own imagination.

The routine for moving the box will be simple. We'll use a sine for the x coordinate and a sine for the y coordinate (it'll look very smooth). We need to avoid accidentally drawing parts of the graphic out of bounds (which can lead to a segmentation fault), so the equations won't look exactly like 160*sin(x)+160 or 100*sin(x)+100, but they'll be fairly close. Sine equations are very useful, and chips these days are very fast at trig so you don't have to worry about optimizing, for example, by building a table of already-calculated sine values. For fun, we could also cycle the colors so that the image would look groovy, but you can try to implement this yourself, if your computer is fast enough. Here we go, and remember, this time we'll need the math library, so compile with

```
gcc -Wall -O2 joyousbox.c -lvgagl -lvga -lm -o\
  joyousbox
```

(recent gcc/egcs versions will automatically link the math library, but to be sure, we like to include it). See Listing.

—Jason Kroll

Listing. joyousbox.c

```
#include <vga.h>
#include <vgagl.h>
#include <math.h>
#include <stdlib.h>
#define GMODE G320x200x256
/* It's a good idea to keep these global */
GraphicsContext *virtualscreen;
GraphicsContext *physicalscreen;
int main(void)
{
  int c, d; /* counting */
  int x, y; /* box location */
  int *box; /* our box */
  int *cut; /* cookie cut */
  vga_init(); /* start svgalib */
  gl_enableclipping();
  vga_setmode(GMODE); /* set mode */
  gl_setcontextvga(GMODE);
  physicalscreen = gl_allocatecontext();
  gl_getcontext(physicalscreen);
  gl_setcontextvgavirtual(GMODE);
  virtualscreen = gl_allocatecontext();
  gl_getcontext(virtualscreen);
  /* now let's fix the palette up
   *  real pretty like!  */
```

```
    for (d=0; d<256; d++)
      gl_setpalettecolor(d, 32*sin(6.3*d/256.0)+31,
                            32*sin(6.3*(d-67)/256.0)+31,
                            32*sin(6.3*(d-133)/256.0)+31);
    /* generate our square, with a
     * transparent hole in the middle */
    box = malloc(16*16*BYTESPERPIXEL);
    cut = malloc(16*16*BYTESPERPIXEL);
    for (d=0; d<7; d++) /* loop to draw box */
      gl_fillbox(d,d,16-2*d,16-2*d, (d-6)*60);
    gl_getbox(0,0,16,16,box); /* get the box */
    /* draw the background stripes */
    for (d=0; d<HEIGHT; d++)
      gl_hline(0, d, WIDTH-1, d+1);
    /* the main loop. For fun, try adding a few
     * more boxes following the leader
     */
    for (c=d=0; d==0; d=vga_getkey()) {
      c++;
      x = 152*sin(c/37.0)+152;
      y = 92*sin(c/61.0)+93;
      gl_getbox(x,y,16,16,cut);
      gl_putboxmask(x,y,16,16,box);
      gl_copyscreen(physicalscreen);
      vga_waitretrace();
      gl_putbox(x,y,16,16,cut);
    }
    return 0;
  }
```

TALKING TO TIM

Using Linux and Informix to register for conferences is a reality; it happened at ALS (Atlanta Linux Showcase) last year. I talked to Tim Costello, the man who made it happen, by e-mail on January 31. Tim is a Senior Systems Analyst for Conference Management Systems, LLC. This company specializes in convention services, e.g., registration, travel and hotel booking, message centers, booth locators, temporary staffing and related services.

Margie: Tell me a bit about yourself.

Tim: I have worked with CMS since April 1994. I have been in the computer industry since about 1984—before that, it was just a hobby—when I started doing some Dbase programming and general consulting for a local company.

When I first started college in 1986, I planned on an EE degree with a CS minor. In 1992, I graduated with a degree in Technical Theatre (Lighting and Sound Design) with a CS minor. After college, I worked as a consultant Monday through Thursday and did lighting for local events most Fridays through Sundays. Then in late 1993, I got a call from a friend who was touring with Disney's "Beauty and the Beast on Ice" in Europe. They needed an electrician ASAP, so for the next six months I toured Europe with Disney. At the end of the tour, I received a job offer from CMS, a consulting client of mine. So the day after I flew back from Barcelona, I was at work at CMS buying equipment for one of our early shows.

Margie: How long have you been involved in the Linux community?

Tim: I first started using Linux early in my tenure at CMS, so I would guess late 1994. I haven't contributed any programming to the community, but I answer questions in the newsgroups whenever I can and submit as many bug reports as possible.

Margie: How did you come to be taking care of ALS registration? Do you belong to ALE?

Tim: We submitted a proposal for registration services in response to a request from ALS. One stipulation in the Request For Proposal from ALS was the use of open-source software, which we were already using—we were a perfect fit. No, I am not part of ALE; CMS is based in Park Ridge, IL.

Margie: Have you done registration of shows before?

Tim: Convention registration is a core business for us. We do registration for several hundred shows a year, ranging from small corporate meetings to ones with over 20,000 attendees.

Margie: How did you come to pick Informix as your database of choice? How did the combination of Linux/Informix work for you? Have you used other systems to take care of registrations? How did Linux compare?

Tim: I'll answer all these questions in a group. When we first started doing registration, we used HP9000s/HP-UX and Informix—all purchased prior to my coming on board. During this time, I was using a Linux box as my office desktop and was the system administrator for both systems. As time went by, whenever we needed a new server, I would use a Linux box; e.g., our dial-in/fax server is a Linux box using **mgetty-sendfax**. Until Informix came around and ported to Linux, I was stuck with the HP9Ks. I was involved in the lobbying effort (I signed the e-petitions) to get Informix to port to Linux, and as soon as it was available

as a public beta, I started testing with it. I find the Linux boxes easier to administer than the HPs, and on a price/performance curve, I find them much more attractive.

Margie: Would you have preferred to use some other system?

Tim: No. We began a move to SCO UNIX as a stop-gap when Informix was still publicly undecided about a port. Those systems were wiped and moved to Linux as soon as possible.

Margie: What do you feel are the main advantages to using Linux/Informix over other systems?

Tim: Ease of system administration and availability of assistance.

Margie: Drawbacks?

Tim: Commercial software availability, or the lack thereof. One of the other services we offer for conventions is a voice-messaging system, using Dialogic voice cards. I am currently forced to use Windows and Visual Basic to access them, and until a month or so ago, there seemed to be no hope unless I switched vendors—something I may still do, but I hate to waste my existing investment. However, when Intel recently acquired Dialogic, they announced they would support Linux!

Margie: Do you use Linux/Informix in other business situations? If so, tell us about them.

Tim: We use Linux for all our servers, web (with Apache), fax (with mgetty), file and print (with Netatalk) and database (with Informix).

Other open-source systems in use at our location are Python, PHP3 and Perl, among others. We use Red Hat and Linux-Mandrake on our servers, with some updates/patches.

Margie: What did you think about the ALS event itself?

Tim: I enjoyed every minute of it! Next year, we plan on additional people from our staff going to the show, so we can trade off working at registration. There were several sessions I would have liked to attend, but missed due to my duties at registration.

Margie: Thanks for your time. —Marjorie Richardson

THE BUZZ

Marc Torres leaving SuSE to head up Atipa. (http://www.linuxjournal.com/articles/briefs/054.html)

Andover.net's acquisition of QuestionExchange, a company offering technical support in an auction-type setting. Ask a question, accept bid, get your answer. (*Linux Today*, January 28)

IBM's big plans for Linux. Big Blue announced its line of network computer terminals can now run on Linux, and it will soon make key Java software components available to leading distributors of Linux. (*Linux Today*, January 31 and January 26)

Sun Microsystems' release of version 8 of Solaris for free. Well, there will be a $75 fee for the bundle of applications that comes with it. (*Linux Today*, January 28)

Kevin Mitnick finally getting out of jail. (http://www.linuxjournal.com/articles/culture/005.html)

Arrest of Jon Lech Johansen in Norway for breaking DVD encryption scheme. (http://www.linuxjournal.com/articles/culture/007.html)

Penguins in sweaters after oil spill. (http://www.phillipisland.net.au/)

SLASH QUOTES

On the first day of LinuxWorld Expo (the perfectly numbered 2/2/2000), I walked into the press room and was greeted by the sight of fourteen PCs, all with browsers open. Half of them were tuned in to Slashdot.

Nothing in the Linux world (Expo or otherwise) is more popular than this site, where CmdrTaco, Hemos, Roblimo and their cohorts feed readers a steady diet of “News for nerds. Stuff that matters.” But Slashdot is a source of news like a fireplace is a source of bricks. In fact, fireplace is a good analogy for the function Slashdot serves in the nerd community. Each news item is a log thrown into the fire. Combustion always follows—dozens to hundreds of comments break out.

Many of the comments either bear quotes or have signatures that are themselves worth quoting. Below are just a few.

—Doc Searls

- If there is a God, you are an authorized representative. —Kurt Vonnegut, Jr.
- 0 1, just my two bits. —Cid Highwind
- Those who will not reason, are bigots, those who cannot, are fools, and those who dare not, are slaves. —George Gordon Noel Byron (Lord Byron)
- When the facts change, I change my mind. What do you do? —John Maynard Keynes
- I'm not as good as I once was, but I'm as good once as I ever was. —Astro Jetson
- Moderation is good, in theory. —Larry Wall
- I can picture in my mind a world without war, a world without hate. And I can picture us attacking that world, because they'd never expect it. —Bad Mojo
- There are three kinds of people: those who can count and those who can't. —Anonymous Coward
- I've lost my faith in nihilism. —hey!
- A year spent in artificial intelligence is enough to make one believe in God. —CrudPuppy

CLUELESS IN TOYLAND

Eric Robison is a UNIX consultant of long standing whose one-man company, Clue Computing, had the good sense to register clue.com as a domain name in 1995.

Hasbro is a toy company of long standing that makes, among hundreds of other products, a board game called “Clue?”.

Like many big old companies, Hasbro was rather clueless about the matter of domain names until it was too late. When they discovered that Mr. Robison had already registered clue.com, they did what comes naturally to many big old clueless companies: they sued him. They also lost. Naturally, they appealed the judgment. So the fight is still on.

When we asked Mr. Robison for a few words about the case, he framed his response in the manner of the “*LJ* INDEX”. With his permission, we reproduce it here.

1. Years Clue Computing has been in this fight: **5**
2. Dollars spent by Clue Computing on the fight: **~100,000 US**
3. Highest advertised domain name sale price in 1995: **~$100,000 US**

4. Highest advertised domain name sale price in 1999: **$7,500,000 US**
5. Total number of lawyers in law firms working for Clue Computing: **2**
6. Total number of lawyers in law firms working for Hasbro: **>1,000**
7. Number of settlement offers made by Clue Computing: **>5**
8. Number of settlement offers made by Hasbro: **0**
9. Closing stock price for Hasbro (HAS) for the week of 6/2/95: **35 1/4**
10. Closing stock price for Hasbro (HAS) on 1/31/2000: **15** (the boycott must be working).
11. Cost of one share of Clue Computing, since its founding: **$10** (okay, so we're not publicly traded...)
12. Number of domain names Hasbro had registered in 1995: **about 20**
13. Number today: **probably over 100** (**whois** dies after 50, and they had over 60 before NSI turned off whois a few months ago. Hasbro also tends to hide their registrations under false names and third parties.)
14. Number of domain names Hasbro wishes to register: **thousands**, one per product or service they sell
15. Hasbro's management: **the stupidest SOBs in the universe**
16. Clue's management: **the stubbornest SOB in the universe**

—Doc Searls

THE AMIGA LIVES ON

The world's first multimedia computer (and what a computer it was, as any Amiga freak would love the chance to tell you) came out in 1985 with a 7.1MHz MC68k, 4096 colors, stereo sound, and was enough to convert even Commodore 64 fanatics. Oddly enough, Commodore acquired Amiga for $40 million back in July of '85, and then destroyed its hopes of success by hapless marketing ploys. Commodore itself died in 1995, and was sold to ESCOM for $12 million. Gateway ultimately acquired what remained of Amiga, and it looked as though they were poised to bring back the once and future queen. However, Gateway had bought Amiga for the patents, not for its devotees, and the new vaporware Amigas never materialized. (You remember the rumors—that it would be Linux-based, etc.)

Today, Amiga is owned by Amino Development Corporation which has changed its name to Amiga Corporation and plans yet again to resurrect *the* multimedia machine. However, *you* don't have to wait; Amiga will carry on. Rush on over to http://www.themes.org/ and you can download Amiga themes for your favorite window managers. Yes, that's right, the Amiga Workbench 2.*x* series (known as Picasso, presumably because the colors correspond with Picasso's blue period) can live again on your desktop. Pair it with GNOME and its new anti-aliasing

feature, and you've got it. Fire up the GIMP, start up an ETerm and maybe load XClock; it'll be just like home. Well, until you try to load Video Toaster. But, hold on a while; what with DVDs, MP3s, a bit more sound support and the new wave of games (and the hardware support they will drive), we might actually have multimedia machines someday.

—Jason Kroll

THEY'RE AT IT AGAIN

Last month in my games column, I noted that the authoritarians had given up the MP3 war and lost control of the DeCSS situation. Unfortunately, some people don't know when to quit, and the industry is back at it again. This time, the Recording Industry Association of America has held secret meetings in Seattle in order to plot the demise of MP3 as well as other conniving ways to take control of technology out of the hands of users. The real news, however, was that the Motion Picture Association of America got in hot water over the raiding of DeCSS author Jon Lech Johansen's house by special police. The hacker community more or less collectively called for a boycott, ranging from DVDs to the entire motion picture industry and even extending to every single product, film-related or not, produced by any of the big seven (Disney, Sony, MGM, Paramount, Fox, Universal Studios and Warner Bros.). The 2600 community organized large protests outside movie theaters across the country. This charming graphic featured prominently in the thousands of flyers we distributed.

—Jason Kroll

TUX GAMES

Enthusiasts of Linux gaming, and those who realize games drive the hardware industry, will be pleased to hear that Tux Games (http://www.tuxgames.com/) is shipping its new Demo CD of six playable demos of popular Linux games for $7.50 with free shipping. Titles include Loki's Civilization: Call to Power, Eric's

Ultimate Solitaire, Heroes of Might and Magic III, Myth 2: Soulblighter and Railroad Tycoon II, as well as a playable demo of MP Entertainment's Hopkins FBI. Yes, Linux has games, did you know? Check out http://www.happypenguin.org/ and http://www.linuxgames.com/ for general information on the state of the art in Linux gaming.

—Jason Kroll

LJ INDEX—APRIL 2000

1. Linux market share among Intel servers, as projected in a recent IDC report: **24.8%**
2. Size of first Linux kernel: **71KB**
3. Lines of kernel code in v2.0: **3/4 million**
4. Number of penguins killed in the recent mystery spill near the Phillip Island Nature Park: **19**
5. Amount of oil discharged in the Bass Strait: **1000 litres**
6. Number of penguins rescued: **205**
7. Amount of money donated to the Phillip Island rescue efforts by Linux users: **$5,000 US**
8. Number of the 25 species seriously affected by the 1989 Exxon Valdez oil spill that have recovered: **2**
9. Amount of the $5 billion US Exxon was ordered to pay in punitive damages, that it *has* paid: **0**
10. Number of years Kevin Mitnick spent in jail awaiting trial: **5**
11. Number of e-mails received by Jason Kroll, in response to his on-line tome, “Free Kevin, Kevin Freed”: **77**
12. Number of e-mails received by Mr. Kroll in response to his recent on-line article “Crackers and Crackdowns”, about the DeCSS fiasco: **623**
13. Number of subscribers to PursuitWatch, an L.A. paging service that alerts customers when a high-speed chase is televised: **350**
14. Number of days for the longest up time on a Linux system: **498**
15. Amount of sales for Corel Linux in 1999: **$3.2 million US**
16. Percentage of embedded application developers who plan to use Linux as their host platform in the current year: **26**
17. Percentage of IT professionals who consider Linux important or essential to their enterprise strategies: **49**
18. Percentage of corporate PCs that will run Linux and other free operating systems within two years: **9**
19. Current Linux corporate PC market share percentage: **4.5**

20. Percentage of enterprises that plan to deploy Linux or FreeBSD as corporate e-commerce platforms: **17**
21. Percentage of IBM servers of all sizes that now support Linux: **100**
22. Number of IBM server lines that supported Linux one year ago: **1**
23. Worldwide open-source UNIX (including Linux) growth rate for major enterprise server applications: **150% to 500%**

Sources

- 1, 15: *LinuxToday*
- 2, 3, 14: http://www.pelourinho.com/linuxatlax/linuxtrivia/index.htm
- 8, 9, 13: *Harper's Magazine*
- 4-7: Phillip Island staff, http://www.penguins.org.au/
- 10-12: Jason Kroll
- 16: Electronic Market Forecasters
- 17: MERIT Advisory Council
- 18-23: Survey.com

STRICTLY ON-LINE

Web Analysis Using Analog by Gaelyne R. Gasson is an introduction to this open-source program. Analog analyzes log files from your web servers and gives you many different reports, in your language of choice (it supports 35). Find out how you can obtain accurate statistics on web traffic to your sites with this easy-to-use program.

Shell Functions and Path Variables, Part 2 by Stephen Collyer is a continuation of the series that started in the March issue. This time, he takes a detailed look at the **addpath** function and how it is used.

Enlightenment Basics by Michael J. Hammel is a guide to getting and installing Enlightenment. It is an excellent precursor to Mr. Hammel's article in this issue on using Enlightenment (“Artists' Guide to the Desktop, Part 2”).

The Generation Gap by Brian R. Marshall is a serious discussion of the issues involved with the use of open-source software components in closed-source applications. Giving the pros and cons of this controversial subject, this article is not to be missed by those interested in the Open Source movement and all its ramifications.

HARBINGER

From the beginning, Linux has been something of a hermit crab operating system, because it tends to inhabit boxes designed first for other operating systems. This has been especially true for clients. While special-purpose servers have been built around Linux for years, clients have mostly been Window boxes with Linux flowers, instead of the more familiar sort.

Portables have been especially vexing to Linux hardware manufacturers. All your familiar laptops are packed with arcane drivers and embedded characteristics that make running Linux somewhat of an iffy proposition.

Not any more. Now we are seeing a new generation of portables designed from the ground up to run Linux. One of the first out of the gate appears to be a remarkable new machine from Boxx, http://www.boxx.net/. Described as “the first portable/slim desktop hybrid computer designed from the ground up for Linux-compatible multi-platform computing”, it's a veritable arsenal for the road warrior.

Despite its extreme variety of physical features, its best talent may be its dual-boot capabilities. The user can install and run two x86-compatible operating systems, one off the primary hard drive and the other off the swappable device bay, key-selecting between the two—it's like having two computers in one.

Here are a few more features of Boxx computers:

- Convertible from notebook to slim desktop, presentation easel and pen tablet configurations
- Detachable wireless (IR) keyboard and wireless entertainment remote control
- 14.1 or 13.3-inch TFT XGA LCD screen with resistive touch-sensitive panel (laser-pointer pen stylus)
- Swappable device bay allowing a second HDD, CDRW, DVD ROM, CD-ROM, LS 120, FDD or battery
- 3-D stereo sound with built-in active diaphragm subwoofer
- Power system with three batteries (for up to 12 hours operating time)
- Available in summer 2000

—Doc Searls

DISTRIBUTION WATCH: Linux on PowerPC

In a recent article, three flavors of Linux that work on PowerPC were listed, including NetBSD (often not recognized as a flavor of Linux, and for good reason—it isn't one!), MkLinux (an implementation that sits on top of the Mach kernel) and LinuxPPC (a typical Linux distribution for PPC). Three species? Diversity is a big evolutionary advantage, and Linux intends on stickin' around, so maybe you can see what's coming up 5th Avenue.

Linux for PowerPC is available from a number of sources, the latest of which is SuSE. The chameleon enthusiasts from Germany have delivered a beta of 6.3, and once the kinks get ironed out, we can look forward to lizards on our Apples. TurboLinux, even though it does not have a cute mascot of any kind, is nevertheless able to bring its Japanese, Chinese and English language distribution to users of the Motorola. Getting back to fuzzy furry animals, Terra Soft's Yellow Dog Linux is yet another offering for Apple PowerPCs and IBM RS/6000s.

- PowerPC Linux resources: http://ppclinux.apple.com/
- SuSE Linux: http://www.suse.com/
- TurboLinux: http://www.turbolinux.com/
- Yellow Dog Linux: http://www.yellowdoglinux.com/
- NetBSD Project: http://www.netbsd.org/
- MkLinux.org: http://www.mklinux.org/
- LinuxPPC: http://www.linuxppc.com/

—Jason Kroll

STOP THE PRESSES: Corel and Inprise Merge

A year ago, the term “Linux business” was something of an oxymoron. Business? With “free software”? Well, that was before the Red Hat IPO, which was when the world changed. By the time 1999 was out, “Linux business” included VA Linux, Andover.net, Cobalt and other relatively new publicly traded companies whose combined market value totaled in the dozens of millions of dollars.

Suddenly the question was, “Who are these guys going to buy with all this new market cap?” Red Hat started with Cygnus. VA started with Andover (which had its own spectacular IPO at about the same time as VA's). All eyes turned toward a pair of well-established PC software companies that had recently repositioned themselves as Linux businesses: Corel and Inprise/Borland.

Both looked like they would help fill out the product portfolios of either Red Hat or VA Linux. Neither Corel nor Inprise would be easy to buy: both were profitable, with revenues in the hundreds of millions despite relatively depressed market stock values. But it looked do-able.

Then, on February 7, the two companies did something no one expected: they merged with each other. The new company looked to boast total sales of over $400 million, profits of a quarter that sum, and a market cap at merger of $2.44 billion. It would be called Corel, finally retiring Inprise/Borland's identity crisis (though key Inprise products would still bear the Borland brand).

Olive branches were immediately extended to the rest of the Linux community. In an interview with *Linux Journal*, Dale Fuller (the Inprise interim president and CEO who will become chairman of Corel's board of directors) declared his intention to partner with all the other Linux players, as well as its growing legion of developers. “We want to work with all the distributions and with all the development communities,” he said. “They'll all need applications and development tools. That's our business. We're here to help.”

The true test will come when Corel finishes coming out with its complete office application suite, and Borland's Kylix project completes the Delphi and C++ Builder development products for Linux. With all those products ready, Linux will become truly competitive (as well as compatible) with Microsoft on the desktop. Will customers buy it? Stay tuned.

—Doc Searls

VENDOR NEWS

DataViews Corporation announced an open-architecture HMI software development tool for Linux, allowing developers to create highly customized user interfaces for the monitoring, control and simulation of dynamic data.

In cooperation with **SuSE**, the developers' group at **Hans Reiser and Chris Mason** have expanded the high-performance ReiserFS (file system) with a journaling function. The release for SuSE Linux 6.3 can be downloaded from ftp.suse.com:/pub/suse/i386/update/6.3/reiserfs

Corel Corporation announced its Corel LINUX OS desktop will be able to run Windows applications seamlessly over any connection. A release containing both Linux client and Windows NT server licenses for **GraphOn Bridges** is scheduled to ship in mid-2000.

The Linux Professional Institute (LPI) announced the immediate availability of the first exam in its Linux certification program. The exam, which covers Linux basics as part of the program's first level, is now available worldwide at testing centers affiliated with Virtual University Enterprises (VUE).

TSCentral, the Internet's most comprehensive business and professional event directory, has launched a new industry section devoted to Linux at http://www.linux.tscentral.com/.

QLogic Corporation, a provider of Fibre Channel host bus adapters, and SCSI connectivity solutions, announced it has purchased AdaptiveRAID from Borg Adaptive Technologies, Inc., a wholly owned subsidiary of nStor Corporation.

Microsoft and **Caldera** announced they have reached a mutually agreeable settlement of an antitrust lawsuit filed by Caldera in July 1996. The terms of the agreement are confidential.

Red Hat, Inc. announced the appointment of Michael Tiemann to the position of Chief Technical Officer. Tiemann is replacing Marc Ewing, co-founder and former CTO of Red Hat. Ewing will remain as a member of the board of directors of Red Hat, Inc.

To meet the growing demand for Linux solutions for the enterprise, **Atipa Linux Solutions** announced the opening of three new offices in New York, San Francisco and Austin.

Knox Software, a supplier of backup solutions for Linux, announced that Arkeia 4.2 has earned IBM Netfinity ServerProven1 Solution validation.

MontaVista Software Inc., developer of the Hard Hat Linux for embedded computers, announced it has hired Kristin Anderson as director of support.

Corel Corporation announced it has entered into an agreement to acquire up to a 30 percent stake in OE/ONE.com, a start-up company developing an "Information appliance" platform or thin-client Internet Appliance platform.

Digi International announced an agreement with Red Hat, Inc. to join in marketing programs that will enable distributors, resellers and integrators to offer Linux-based communications servers designed specifically to suit the needs of small- to medium-sized businesses.

A new strategic partnership has been established between **Minolta** and **SuSE** to facilitate printer support. Linux users can now enjoy high-quality output from the Minolta PagePro 8, 18 and 25 monochrome laser printers.

SpellCaster Telecommunications Inc. has acquired **MediaGlobe Networks**, giving it full ownership rights to a Linux-based server software family.

iNUX Inc. announced the release of a small business desktop computer utilizing Linux to deliver easy and intuitive access to a broad selection of pre-loaded and pre-configured applications and content.

Linsight, an on-line Linux information resource, announced E. J. Wells is now its co-director. Dave Whitinger, founder, believes promoting Wells within Linsight, which already provides an authoritative resource of all Linux upcoming events and available Linux training, implements a new beginning.

Collab.Net, provider of scalable services and infrastructure for the development of open-source software, announced the appointment of James Barry as Vice President of Strategic Initiatives, Frank Hecker as Systems Engineering Manager and Jason Robbins as Senior Software Engineer.

At the **PHP Developers' Conference** last week, the PHP Group made a number of decisions and established working guidelines for the continuation of code development for PHP.

SpellCaster Telecommunications Inc., a developer of ISDN and remote access technology for Linux, announced it is releasing the source code for its Babylon software under the GPL. Babylon provides point-to-point remote access to and from Linux systems using PPP.

Creative Computers announced the change of its name to IdeaMall and the launch of eLinux.com, which will provide products from distributors, multi-

vendor Linux solutions and custom configurations of Linux systems through a secure web site.

Atipa announced that Marc Torres is now the Chief Technology Officer of Atipa Linux Solutions. He brings to Atipa thirteen years of multi-platform UNIX experience, including specialties in Network Management and Systems Architecture, having previously served as President of SuSE Inc.

GraphOn Corporation announced it is providing the National Research Council of Canada with GraphOn GO-Joe connectivity software to enable researchers around the world to access the hundreds of applications and databases at the Canadian Bioinformatics Resource over the Internet and Canada's CANARIE Optical IP advanced network, CA*net 3.

O'Reilly Network announced the launch of its technical portal, http://www.oreillynet.com/ and its new Linux DevCenter, http://oreilly.linux.com/.

Netizen, a Melbourne-based IT consultancy specializing in open-source software, announced it will be offering system support contracts for Linux, FreeBSD and other open-source systems for Australian clients.

Alpha Processor, Inc., a developer of high-performance Linux solutions, and **Red Hat, Inc.** announced a technical partnership to create a world-class center for the development of Linux clustering technology. The new clustering facility will be located in Research Triangle Park, North Carolina.

Factoid: Penguins' tongues are covered with many little spiky spines that all point backward into the throat, so that when a penguin catches a fish, it is gripped by the spines and cannot escape.

Trivia: What was the first Linux distribution to be certified Posix-compliant? Lasermoon Linux FT

Rumor: *Linux Journal* is being sold—and it's just that, a rumor.

Link of the Month: www.gnu.org/brave-gnu-world/brave-gnu-world.en.html

Archive Index Issue Table of Contents

Advanced search

Where Have the Nets Come From?

Peter H. Salus

Issue #72, April 2000

As of the end of last year, there were over 68 million hosts attached to the Internet, and those hosts average four users each.

On September 2, 1969, the first piece of the ARPANET was installed in Len Kleinrock's lab at UCLA. A month later, the second went into Doug Engelbart's at SRI in Menlo Park, CA.

As of the end of last year, there were over 68 million hosts attached to the Internet, and those hosts average four users each. A quarter of a billion folks all over the world.

I really do mean all over.

The latest analysis by MIDS (http://www.mids.org/) of the data assembled by Network Wizards shows host sites in over 200 countries or territories—far more than the members of the UN.

MIDS noted a year ago that while the Internet was growing, the growth rate, which had shown a factor of two or better (i.e., the size of the Internet doubled each year) for just over a decade, was beginning to slow. The four most recent data points (January 1998, July 1998, January 1999, July 1999) show the growth rate to be 1.5 now. This is still a lot.

The 68 million of January 2000 is projected to become 100 million this year, 150 million at the end of 2001, and 225 million at the end of 2002.

If we assume the average will stay at four, that's a billion people in less than three years.

Why Has the Growth Been So Incredible?

I think there are several factors: first, the advent of the Hayes modem. The progress from 1969 to 1975 brought us from the half-rack that was used in the original ARPANET to the shoebox of the acoustic modem. The subsequent reduction in size (and cost) was vital. Second, there was the “miracle” of the desktop computer—the Apple, the LISA and the IBM PC (then the XT and the AT). I'm not trying to be detailed or expansive here: I know about the Osborne and the Kaypro, etc.

With these two developments, you could have a machine at home, plug it into your phone line and communicate with others. That is, if your line had modular plugs; otherwise, you either bought a plug with four circular pins on one end and an RJ-11 on the other, or you wired into your line at the wall and attached a cable with an RJ-11 at the end—both were “illegal” acts where Ma Bell was concerned.

In October 1981, there were 213 hosts on the Net. In February 1986, there were 2308. From 1986 to 1997, the number of hosts doubled every year (the actual slope of the line is 2.176). But since January 1998, as I noted above, that growth rate has fallen.

I wouldn't worry about this.

Every technology goes from a lab to being the toy of an in-group to being a utility. When it first becomes a utility, people rush to buy and participate, then there is a slackening.

But don't fret. In July 1999, Malawi, Kiribata, Eritrea and Somalia joined the Internet. In January 1999, they were preceded by Sao Tome, Samoa and Tuvalo. There are so few additions because there are so few countries left unconnected.

The scale of that penetration and interconnection is quite important. MIDS' analysis of the July 1999 data lists over 250 geographic entities. I don't use “countries”, because Guam, Antarctica and the Vatican City (for example) aren't countries. But that's still more places on the Internet than countries belonging to the United Nations.

Doc Searls pointed out some months ago that about a third of the Net servers were powered by Linux. As the Internet grows, Linux's use will grow with it. There's no way that Windows (which comes in third, after UNIX and Linux) will overtake Linux in the Internet back rooms—nor, I imagine, in the corporate intranets.

I'd put my money on technology by Torvalds, Gosling, Ousterhout, Allman, Wall, et al., from companies like Transmeta, VALS, Red Hat, Caldera; not on Wintel.

Peter H. Salus , the author of *A Quarter Century of UNIX* and *Casting the Net*, is an *LJ* contributing editor. He can be reached at info@linuxjournal.com.

Archive Index Issue Table of Contents

Advanced search

Now What: Are We Going to Let AOL Turn the Net into TV 2.0...

Doc Searls

Issue #72, April 2000

AOL Time Warner makes Microsoft look like the corner store. --John Katz

The AOL/Time Warner deal is the “News that Won't Go Away”. Almost overnight, Microsoft became a non-issue. Some commentators even began to see Bill Gates as a sympathetic figure, like Caesar in his last days. Suddenly, we had new Caesars to worry about: AOL's Steve Case and Bob Pittman. The feds could shatter Microsoft into a dozen Baby Bills (or Baby Borgs), and everyone would still be wondering if AOL could actually pipe-weld the Net to the rusty back end of TV's history.

Break up Microsoft? Like, so what? All the pieces will still be caught in the Net, and more than half the people who log on have @aol.com addresses. Which is creepier: a crashy OS, or a giant ISP that dumbs down the Net?

Well, are we going to do something about that? Can we? Slashdot's Jon Katz thinks the situation could hardly get worse. “There is no way for innovators or entrepreneurs to challenge it,” Katz writes about the AOL/Time Warner deal. “No one can compete with the software, the cheap access, the wide range of content. This company is bigger than many countries.” How big, exactly? Let's take a look at what Time Warner brings to the table:

- Publishing: Time Inc. has 33 magazines, including *Time*, *Life*, *People*, *Southern Living*, *In Style*, *Sports Illustrated*, *Parenting*, *Money*, *Health*, *Sunset* and *Fortune*. *Time* alone accounted for more than 1/5 of all revenue generated by consumer magazines in 1998. Little Brown, Sunset Books and other Time imprints account for dozens of best-selling books every year.

- Cable: CNN, TBS and HBO each have multiple channels, enormous production operations and massive libraries of programs. Then there is Time Warner Cable, which serves 21.3 million homes.
- Motion pictures: Warner Brothers has 5,700 feature films, 32,000 television shows and 13,500 animated titles. New Line Cinema is one of the top studios as well.
- Music: Warner Music Group accounted for about 1/4 of 1998's top-selling albums.

Meanwhile, AOL sells the Internet on training wheels to nearly 55% of the people who log on from the U.S. alone, and they make money doing it—more than $5 billion US a year. That's on top of Time Warner's $26+ billion US.

Against that, we've got what? A bunch of hackers? Well, yeah—and that's why we'll win.

AOL didn't invent the Net. Hackers did. They didn't invent it as yet another way to pump out *content* and suck back credit card purchases. They invented it so no one could own it, anyone could use it, and nothing could threaten it.

“All the significant trends start with technologists,” Marc Andreessen told us back in 1998, when we interviewed him about the open sourcing of Netscape's Mozilla code (“Betting on Darwin”, *Linux Journal*, August 1998). In the same interview, he added, “Technologists are driving progress, and it's easier to drive with Linux than with anything else.”

So let's savor this irony: while Netscape now belongs to AOL, and Mozilla (still funded by what's left of Netscape) lags behind Microsoft's Internet Explorer in the consumer market, Intel is quietly building its new home Internet appliances —TV set-top boxes and thin network clients—with Mozilla code running on Linux. If these catch on, they'll bypass AOL and every other mass-market megalith that continues to regard the Net as yet another one-way shipping system between a few suppliers and a zillion consumers. These appliances will prove yet again that the connections which matter most are the ones between human beings, and that includes the human beings who do e-business with each other.

From the day the first packet moved across a TCP/IP network, economic power has been shifting steadily from supply to demand. Wars and marriages between giant suppliers still make great stories, but those stories have little or nothing to do with what's really going on. Hackers—the programmers, inventors, developers and architects who are building this new world—have been trying to make sure the stuff that matters most is what works for everyone because it belongs to no one. They do it by turning markets into what

they were thousands of years before industry turned “market” into a verb: places where people gather, talk about what matters to them and do business together.

Demand will win because it is equipped to win. Mouse-to-mouse, link-to-link, page-to-page, e-mail-to-e-mail, voice-to-voice, customers are going to come out on top, along with the companies who make it easy to do business with them. Those companies will know that the best way to relate to customers is as human beings, not as abstract populations to attack, control, capture and herd like dumb beasts.

The real war is between markets and marketing. For decades, marketing has been the military wing of business, working “strategically” to “attack”, “capture” and “deliver impact” to populations it calls “eyeballs”, “seats”, “end users”, “demographics” and “consumers”. (Jerry Michalski calls them “gullets that live only to gulp products and crap cash.”) The problem with marketing is that there is no demand—no market—for its insults. That's why markets will win.

Is the AOL/Time Warner deal a bet on markets or on marketing? Credit where due: AOL has done a terrific job of equipping demand to deliver clues (as well as money) to supply. But the consumers AOL wants to “aggregate” and “deliver content” to will only become better equipped to screen out unwanted content, and more significantly, to converse with its sources.

What happens when the mute buttons on remote controls send “we hate this” messages directly back to the advertisers who pay for the media? What happens when consumers turn into real customers with real names who express no desire for *messages* mostly intended for someone else? The business model for mass media advertising falls like a bad tent, that's what.

There's an old advertising adage that says, “I know half my money is wasted. I just don't know which half.” In the advertising tradition, even that's a lie. Direct mail, one of the most efficient forms of advertising, counts a 3% response rate as a success. The dirty truth about most advertising is that it has always been woefully inefficient, especially in mass markets. However, much of it has been successful, which is why it's still around.

That success, of course, came in the absence of alternatives. Worse, it came in the absence of demand from consumers. But consumers were never advertising's real market; they paid nothing for advertising's goods, and exerted no direct influence over it. As a result, countless marketers and “creatives” in advertising agencies (including the hip new “interactive” agencies) still labor over screens and keyboards to come up with *messages* to deliver to people who have little or no interest in it.

The notable exceptions, of course, are classifieds, yellow pages and trade publications like *Linux Journal*, which are sources of both useful editorial content and relevant information paid for by companies of interest to readers.

So here's a clue for mass marketers who think AOL's customers are going to sit still for the kind of heavy abuse that television has been delivering to its addicts for decades: there is no demand for messages. There never was. When that clue finally arrives, it will be like a fist through the screen.

Provided, of course, that the hackers keep doing their good work.

The fight is far from over. The good guys will win the OS war and maybe even the browser war, but there are other enabling technologies that still belong to suppliers with controlling intentions. Streaming media is one. Instant messaging is another. Both could be far more useful than the bait-for-advertising vehicles we see today. Look at the differences between AOL's Instant Messenger and what's starting to come from the Jabber people. Better yet, join the movement. Let's start to show these guys what's really valuable.

Doc Searls is Senior Editor of *Linux Journal* and co-author of *The Cluetrain Manifesto*.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #72, April 2000

Our experts answer your technical questions.

Partitioning and Cylinders

I am installing Red Hat Linux 6.0 on a partition. When I try to make my partitions for Linux, it gives me an error saying “boot partition too big”. I tried to use **fdisk** and it says my hard drive has 1655 cylinders, not 1024, and it will cause problems. How can I get around this? —Will Brown, willj3@mindspring.com

Create a primary partition (the first one) with 10MB; its mount point should be /boot. This is the directory where all the boot images reside, and it needs to be under the 1024 cylinder. The root partition (/) can then be any size, and will not have any problems. —Paulo Wollny, paulo@wollny.com.br

Most PC bioses are still unable to access data beyond the 1024th cylinder. This prevents LILO from booting a kernel that's lying beyond the 1024th cylinder. To get around this, short of getting an alternate architecture without all those ancestral PC limitations, you need to have your boot partition (either /boot if you have one, or your root partition otherwise) completely under the 1024th cylinder. —Marc Merlin, marc_bts@valinux.com

Home Networking

I am setting up a small LAN at home with a server running Red Hat 6.1 and a workstation running Windows 98. I've set each up with a static IP address (192.168.1.1-2) and a netmask of 255.255.255.0. My problem is that I can't even ping my Linux box using the host name. I can ping it using the IP address, and I can ping my Windows 98 box from the Linux box using either the host name or the IP address. I'd appreciate any suggestions you may have.

I had an older version of Linux (Red Hat 5.3, I believe) on the server, and it was working perfectly. I've set up the 6.1 version the same way and it won't work. —Greg Wright, grewright@netzero.net

You didn't mention whether you've configured DNS services at home, so I'll assume you have not. In that case, you need to add an entry to your c:\windows\hosts file. List the IP address of the system you are trying to ping, followed by the host name you wish to use in your ping test. Be sure you do not include the domain name in this entry. You can copy the file hosts.sam if you need a sample. —Chad Robinson, Chad.Robinson@brt.com

Office Networking

I am using SuSE 6.3 and had two networks, one at the main office (192.168.1.0) and the other at the branch (192.168.2.0). I used two Cisco 800 routers to connect them. I can use TCP/IP normally (ping, ftp, http from one host to another). My problem is I cannot see the remote PCs in Windows Explorer at the office. I configured a Linux server with DNS, mail and FTP, but it still doesn't work. At the office, I'm running IPX (there is a Netware 3.11 server) and TCP/IP protocols. Someone told me I must set up a Windows NT running WINS service. I don't want to install Windows NT. Any ideas? —Carlos Germán Siufi, csiufi@puntoar.net.ar

This question involves Windows networking, and more specifically, netbios name lookup. Netbios, which is at the core of Windows networking, was very badly designed and grew from there. While UNIX uses DNS for name lookup, Windows originally used a complicated name lookup scheme based on the election of a local name browser on your subnet and broadcast queries for each name lookup. Because it is broadcast-based, it doesn't work across subnets (which is your configuration), and it was “improved” with the addition of WINS.

The typical solution is indeed to use a WINS server, which is some kind of dynamic DNS equivalent, although it isn't as robust. However, you do not need to run Windows to provide WINS service; Samba (http://www.samba.org/) can do this just fine. While using a WINS server would typically require you to configure each Windows machine, Samba can function as a proxy between local broadcast queries and a remote WINS server. You ought to grab Samba and read its documentation for more info. You may also want to consider getting one of the Samba books. —Marc Merlin, marc_bts@valinux.com

Setting Display Mode

I need to know how to set my display mode when I start the X Window System. It always runs X at 320x300 or something really huge and I can't do anything. I use—Jim, jim@stat.net

Red Hat's X configuration tool is called **Xconfigurator**. If it doesn't work for you, then you should first check that your graphics card is supported by your version of XFree86 at http://www.xfree86.org/#resources/. It may be that your video card is supported by a later version of X, in which case you should upgrade. If your video card is recent, it is most likely Vesa 2-compatible, and you can use the VesaFB server: http://www.xfree86.org/FAQ/#FBDev/. —Marc Merlin, marc_bts@valinux.com

Ethernet Cards

I want to convert my existing proxy server for my home network to Linux. But every distribution I have used has never detected both network cards until SuSE 6.3. I can manually get the card on the LAN to find the DHCP server, and the NT gives it an IP address, but can never ping that server; if I change it to a static IP address, it pings the NT server fine. The real problem is I have a cable modem on the other NIC, on which I have set up DHCPclient. I have viewed and edited the sbin/init.d/dhclient file to make sure it has the **ifconfig $NETDEV 0.0.0.0 up** statement in it. I have read the FAQ and I still cannot get this to work. —Stephen Heaton, srheaton@mediaone.net

All recent distributions work more or less the same in regard to more than one Ethernet card if they are PCI, as they can all be autodetected easily. You may, however, need to add an alias for eth1 in /etc/conf.modules. For example, your conf.modules could look like this:

```
alias eth0 tulip
alias eth1 eepro100
```

After that, you simply need to ifconfig each card in the usual fashion.

Are you sure that Linux does receive a correct IP, netmask and broadcast from the DHCP server? You should type ifconfig eth0 (or eth1) and compare the info you're getting from DHCP with the info you're setting by hand.

The last part, I'm not too sure about. I myself haven't had much luck with SuSE's DHCP client, while Red Hat's dhcp client has worked better for me with no special configuration required. —Marc Merlin, marc_bts@valinux.com

Resources

New Products

Ellen M. Dahl

Issue #72, April 2000

Active Worlds Servers, Atipa Monolith Firewall, Hard Hat Net CompactPCI and more.

Active Worlds Servers

Activeworlds.com, Inc. released Red Hat Linux versions of its Active Worlds servers. In addition, Activeworlds.com will be providing full support to Linux users for its product lines consisting of World Servers, Galaxervers and Uniservers. Active Worlds technology provides high-bandwidth server hosting, backed up by quadruple T1 connections, and allows thousands of simultaneous users to build and explore 3-D content on the Internet in the same shared environment.

Contact: Activeworlds.com, Inc., 95 Parker Street, Newburyport, MA 01950, 978-499-0222, 978-499-0221 (fax), sales@activeworlds.com, http://www.activeworlds.com/.

Atipa Monolith Firewall

Atipa announced a new Linux-based rack-mount firewall appliance, aimed at organizations wishing to secure their web server or internal network. The Atipa Monolith Firewall improves upon traditional “static-packet filtering” by using “dynamic-packet filtering” software that actually inspects each packet for validity before allowing it network access, creating additional security from outside attacks.

Contact: Atipa Linux Solutions, 6000 Connecticut, Kansas City, MO 64120, 800-360-4346, 816-920-6235 (fax), sales@atipa.com, http://www.atipa.com/.

Hard Hat Net CompactPCI

MontaVista Software Inc. released its Hard Hat Net CompactPCI backplane networking package. This release provides embedded Linux application developers with networking options for the CompactPCI designs prevalent in the telecommunications and internetworking markets. It enables both CompactPCI system controllers and peripheral devices to communicate using standard networking protocols across the CompactPCI backplane at high data rates. It supports a variety of protocols, including IP (Internet protocol), IPX, AppleTalk and others available for Linux.

Contact: MontaVista Software Inc., 490 Potrero Avenue, Sunnyvale, CA 94086, 408-328-9200, 408-328-9204 (fax), sales@mvista.com, http://www.mvista.com/.

EZTerm v1.4

CSV Technologies released a new version of its EZTerm software for Linux. EZTerm is a complete Linux command reference designed to eliminate the complexity and drudgery of typing in many Linux command lines. It allows a user to build and modify commands on the fly simply by pointing and clicking. EZTerm gives the user access to a fully functional Linux terminal that includes step-by-step help with the Linux command structure and syntax. EZTerm Software is packaged in the new Red Hat Linux 6.1, and will run with all current Linux distributions including Red Hat, Caldera and SuSE.

Contact: CSV Technologies, Inc., 303-1113 Blanshard Street, Victoria, BC V8W 2H7, Canada, 250-386-4689, http://www.csvtech.com/.

Maple 6

Waterloo Maple Inc. announced Maple 6 for Windows, UNIX, Macintosh and Linux. Maple 6 embodies technological enhancements and new functionality that dramatically speeds up complex technical computation projects. The new math engine delivers a tightly integrated suite of symbolic and numerical solvers. The software combines the flexibility and intelligence of Maple's symbolic computation algorithms with the reliability, accuracy and power of the NAG numerical solver.

Contact: Waterloo Maple, 57 Erb Street West, Waterloo, Ontario N2L 6C2, Canada, 519 747-2373, 519-747-5284 (fax), info@maplesoft.com, http://www.maplesoft.com/.

Motif 2.1.20

Software2Go, LLC announced the release of Motif 2.1.20 for FreeBSD, Linux, NetBSD and OpenBSD on the Alpha, Intel (x86) and SPARC platforms. Motif 2.1.20 is Software2Go's second release of The Open Group's Motif User Interface Toolkit. Software2Go Motif is available in both development and runtime distributions.

Contact: Software2Go, LLC, 76 Corral Drive North, Keller, TX 76248, 817-431-8775 (phone/fax), info@apps2go.com, http://www.apps2go.com/.

Network Shell v.3.0

Shpink Software announced Network Shell v.3.0, a powerful Internet and web-administration server. Version 3.0 supports concurrent remote management of multiple UNIX and Windows 9*x*/NT machines from a single UNIX or Windows NT administration station. Network Shell provides a shell and Perl environment allowing users to perform secure, automated and/or interactive system administration of remote hosts without needing to use TELNET or establish a remote shell connection to each host individually. It supports FreeBSD, BSDI and Red Hat Linux.

Contact: Shpink Software, 3612 Santiago Street, Suite 100, San Mateo, CA 94403, 888-492-6867, 650-525-1537 (fax), sales@shpink.com, http://www.networkshell.com/.

PC300

Cyclades Corporation announced a new product for server-based networking. The Cyclades-PC300 is a WAN PCI adapter that supports one or two serial WAN ports for Internet and inter-office connectivity. Initially offered in a model that supports two serial WAN interfaces (RS-232, V.35, X.21) supported under Linux, the PC300 family will also be available in models with built-in DSU/CSU for direct connection to the communication line. The PC300 can replace access routers and connect remote offices using standard PC servers, providing cost and management advantages without sacrificing performance.

Contact: Cyclades Corporation, 41934 Christy Street, Fremont, CA 94538, 800-882-9252, 510-770-0355 (fax), cysales@cyclades.com, http://www.cyclades.com/.

ESP Print Pro v4.0.2

Easy Software Products announced the ESP Print Pro v4.0.2, a complete printing solution for UNIX. It can print international text, Adobe PostScript, PDF, HP-GL/2, GIF(SM), TIFF, PNG, JPEG/JFIF, SGI RGB, Sun Raster, PhotoCD, PBM, PGM and PPM files transparently to over 1600 printers via serial, parallel and network connections. ESP Print Pro is based on the Common UNIX Printing System and provides PostScript and image file RIPs to support non-PostScript printers.

Contact: Easy Software Products, 44141 Airport View Dr., Ste 204, Hollywood, MD 20636-3111, 301-373-9600, 301-373-9604 (fax), info@easysw.com, http://www.easysw.com/.

Progress Version 8.3

Progress Software is shipping its embedded database and other deployment products on the Linux operating system. Progress provides scalable, multitiered Linux support with Progress version 8.3, a comprehensive suite of integrated development tools, application servers and relational database products. Linux-specific products include Progress AppServer, an application server for sharing components across heterogeneous environments, and Progress Enterprise RDBMS for scalable storage.

Contact: Progress Software Corporation, 14 Oak Park, Bedford, MA 01730, 781-280-4000, 781-280-4095 (fax), asksales@bedford.progress.com, http://www.progress.com/.

Time Navigator

Quadratec announced Time Navigator, its new backup and archive server for Linux. Time Navigator is an advanced solution for on-line backup, archiving and restoration of files, databases and application software for UNIX and other platforms. The new Linux flavour runs on inexpensive x86 to SMP platforms and offers high performance, easy and consistent restore and archiving. Independent of the Linux distribution, the software supports Linux kernels 2.0 and 2.2 which use GNU libc5 and libc6.

Contact: Quadratec SA, Parc Club “Orsay-Universite”, 14/16, rue Jean Rostand, F-91893 Orsay Cedex, France, +33-1-69-33-20-80, +33-1-69-33-20-81 (fax), info@quadratec.fr, http://www.quadratec-software.com/.

Quick Restore 2.6

Workstation Solutions announced Quick Restore 2.6, the first centrally administered, enterprise-ready network backup system for Red Hat Linux (5 and 6) Intel x86 servers and clients. Quick Restore on Linux is seamlessly integrated with all other platforms, allowing backup and recovery of UNIX, Windows NT or Network Appliance data to tape libraries attached to Linux servers. Quick Restore 2.6 also supports Network Appliance filers using the Network Data Management Protocol (NDMP).

Contact: Workstation Solutions, Inc., 5 Overlook Drive, Amherst, NH 03031, 800-487-0080, sales@worksta.com, http://www.worksta.com/.

Storm Linux 2000

storm Linux 2000 OPERATING SYSTEM standard edition

Stormix Technologies announced the release of Storm Linux 2000, available in a Standard Edition and a download edition. It is a new distribution of Linux and builds on Debian GNU/Linux. The Storm Package Manager makes it easy for users to maintain their systems. Storm Linux features a graphical installation and system tools, which include modules for maintaining users and groups, setting up networks and installing and removing software.

Contact: Stormix Technologies, Inc., 555 West Hastings Street, Suite 2040, Vancouver, BC V6B 4N6, Canada, 800-STORMIX, 604-688-7317 (fax), info@stormix.com, http://www.stormix.com/.

Tallyman

Akopia announced the availability of Tallyman, a highly customizable yet easy-to-use system for creating and managing an e-commerce web site. It provides a web-based interface to manage a complex database of products, enter new items for sale, and set up a shopping cart, tax and shipping by filling out a few forms. Tallyman is released under the GPL. It can be downloaded for free from Akopia's web site. The Tallyman Developer release is designed for Linux, but can be modified to run on any major server operating system and with any SQL database.

Contact: Akopia, 1520 S 1300 E, Salt Lake City, UT 84105, 801-994-9680, 707-885-3931 (fax), information@akopia.com, http://www.akopia.com/.

Tango 2000 Application Server

Pervasive Software Inc. introduced its Tango 2000 Application Server for Red Hat, Caldera OpenLinux and SuSE Linux. The scalable, reliable Pervasive SQL 2000 database engine is included, providing developers with a low-cost, zero-administration and robust solution for web and e-business applications. This

server can also connect directly to Oracle, as well as to other leading databases through ODBC drivers.

Contact: Pervasive Software, 12365 Riata Trace Pkwy., Bldg. 11, Austin, TX 78727, 800-287-4383, 512-794-1763 (fax), salessupport@pervasive.com, http://www.pervasive.com/.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Web Analysis Using Analog

Gaelyne R. Gasson

Issue #72, April 2000

Every web site needs a way to get accurate statitics—here's a freely available program to give you that information.

One of the many responsibilities of being a webmaster is keeping tabs on the traffic your site receives. This information can help when it comes time to update: you will know which pages are requested most often and which could use a little more promotion.

Analog is an open-source program that analyses log files from web servers, creating as many as 32 different reports. It works on many operating systems, including Linux, and supports 35 different languages. It's extremely flexible and fast. Its author is Stephen Turner, a Research fellow at Sidney Sussex College at the University of Cambridge. His research has to do with mathematical problems in communications networks, so he has a good solid background when it comes to log file analysis.

Analog's versatility can make it seem very complicated; it has over 200 configuration options. The number of choices is also what makes it very useful for hostmasters and webmasters alike.

A common question on the analog-help mail list is, “How do I setup Analog for virtual hosts?”. This article will cover setup for both virtual hosts and individual pages. In addition, I'll describe how to use configuration and script files to get monthly, hourly, daily and specialized reports.

I've used Analog for several years, first on a FreeBSD system; I was a dial-in customer curious about my personal web pages. Later, I used Analog with Win95/98, and now use it on a Linux system as a web administrator. An example of my publicly available log analysis setup can be found at http://vcsweb.com/logs/.

Analog is the type of program that no two users will use in the same way, due to the vast number of options, only the items I have experience with will be discussed.

Myths About Analysing Log Files

Before delving into Analog's configuration files, it's important to understand what you can and can't learn about your site from log files. First, the log files should be considered only as an estimation of traffic. Log references aren't available when web pages have been accessed from either browser or ISP cache files. According to Analog's author, it's impossible to tell the number of distinct visitors or the total number of visits.

> The bottom line is that HTTP is a stateless protocol. People don't log in and retrieve several documents: they make a separate connection for each file they want. Many times, they don't even behave as if they were logged in to one site. Hence, Analog's emphasis on requests, rather than visits.

What we can learn varies, depending on what the web server writes to the logs. In general, this usually includes the computer name (or IP number) requesting information, whether the request was for a page (generally an HTML document) or another type of file, the status of the request, the time and date the request was made, and the number of bytes transferred. If the log file reports it, we can also learn which web site or page referred the visitor to our site and the browser they were using.

Installation

Analog's home page is at http://www.analog.cx/, and it can be downloaded from www.analog.cx/analog4.01.tar.gz. This web site also lists several mirror sites for the file.

Installation is rather painless: simply type **make** while in the analog4.01 directory. Several options can be set in the analhead.h file prior to using the **make** command, such as host or organization name, the log file path and name; I've found it more practical to use the configuration files for these items instead. The only item I needed to change in analhead.h was the directory path of /usr/local/etc/httpd to /etc/httpd so it matched my Red Hat Linux system, which allows me to use it from the /etc/httpd/analog directory.

It should be noted that Analog will work from any directory, as long as it can find its /lang directory. This can be accomplished by adding the language file and path name to Analog's configuration file(s), but for my use, it's easier to

compile the program to the directory in which I want to work. When using the program as a single user, you can compile and use it from your home directory.

As a second step to installing Analog, I highly recommend viewing and bookmarking the local set of Analog's documentation in your favorite web browser. If you use Analog for creating HTML-format reports, this lets you have the documentation open in one browser window while viewing the reports you've generated in another. Analog isn't the type of program for which you can read the documentation from start to finish, create one configuration file and quit. Rather, the documentation provides information on each of its over 200 commands; you experiment to find the settings that suit your preferences.

Log Files

Analog can be configured to use customized log formats, which is a very good thing if you happen to have log files in various formats created by different servers. Even though I've used a number of different servers, I've been able to continue using Analog to analyze new and old log files (of different formats) by listing the type of log format before giving the name and path of the log file. I now use the Apache web server's combined log format, which produces a common log file that lists the referrer and browser information with the log entry for each access. Otherwise, I'd have separate log files, one for the referrer and another for the browser, and would need to include these log files when working with Analog's configuration files.

If you're a hostmaster, you can configure Apache to use a different log file for each virtual host. This keeps the information for each host separate and makes using Analog to analyze your virtual host log files much more straightforward. This is done using Apache's virtual host directive:

```
<VirtualHost vhost1.com>
ServerAdmin webmaster@vhost1.com
DocumentRoot /www/docs/vhhost1.com
ServerName vhost1.com
ErrorLog logs/vhost1.com-error_log
CustomLog logs/vhost1.com-access_log combined
</VirtualHost>
```

Configuration Files

While you can use Analog with just the analog.cfg file to tell it what to do and where to save its report, if you want to create different reports for virtual hosts and individual pages, it's best to use multiple configuration files. Each configuration file serves a different purpose and can be combined with script files containing command-line switches for Analog.

In this scenario, Analog is run not once, but several times; each run creates a separate report. The analog.cfg file includes only a very few base commands

that relate to our main site, not the virtual host sites. When creating reports for virtual hosts, I exclude analog.cfg from being called with the **-G** command-line switch.

The basic arrangement is similar to a pyramid format. All major items are in a master.cfg file to cover the broad category of all virtual hosts on our system. Items relating only to a specific virtual host and their general preferences are in the next tier, and finally, individual page.cfg files are in the last category. This allows me to create specialized setups as needed and still track individual hosts, sites and pages without making major changes.

When Analog is run for a virtual host, the master.cfg file is called first, followed by the master-vhost.cfg (I replace "vhost" with the name of the host when naming the file), and finally, single-page configuration files for separate pages. An example master.cfg file is included here (see Listing 1).

Listing 1

Virtual Host Master Configuration Files

Listing 2

An example vhost.cfg file is shown in Listing 2, and as you can see, it's fairly general, since most of the report formatting and such is handled by the master.cfg file. The vhost.cfg file can be used to create a "total activity" report for the virtual host. The command-line prompt (or script file), shown without paths for clarity, is:

```
analog -G +gmaster.cfg +gvhost1.cfg
  +Ovhost1-total.html
```

The **-G** tells Analog not to use analog.cfg (which is used for the main host only). **+g** is used whenever we use additional configuration files: there's no space between it and the file name. **+O** designates the output file name: it's the letter O, not the number zero.

Individual Page or Site Configuration Files

Single configuration files are used to give the basic information on the files(s) to include in the report (using the FILEINCLUDE command). The **HOSTNAME** and **HOSTURL** directives are the items that will appear at the top of each report after the words "Web Server Statistics for". For individual pages, we use the name and URL of the page rather than the host name or URL. A single-page configuration file can be three or more lines, as shown in Listing 3.

Listing 3

Notice that the log file to use, output file and report-formatting commands aren't included; these items are set either in the master.cfg files or within the script file when Analog is run. This lets me use the same information when creating the daily and monthly reports, even though the two reports are very different.

The **FILEINCLUDE** command causes Analog to search through the logs and retrieve data relating to only the file you've specified. It's a very powerful command, and is normally used in the configuration files for individual pages or sites. It can also be used with a wild card; if I wanted to include all files in the widgets directory, I would use:

```
FILEINCLUDE http://vhost1.com/widgets/*
```

The command line used to create a daily report for this page (all on one line), shown without path information for clarity, is:

```
analog -G +gmaster.cfg +gmaster-vhost1.cfg
+gwidgets.cfg +Owidget.html
```

Monthly Text Report Configuration Files

I mail a monthly report of web stats to a few of my customers who aren't on-line and who have found the daily reports were too long to print and took too much time to edit. To solve the problem and save time, I created month-vhost.cfg files which create ASCII text format reports. The month-vhost.cfg files are used in conjunction with the individual configuration files described above. A sample month-vhost.cfg file is shown in Listing 4. To produce the monthly text reports, **+a** is used on the command line to designate ASCII output:

```
analog -G +gmonth-vhost1.cfg +gwidgets.cfg\
+Owidget.txt +a
```

Listing 4

Server Activity Configuration File

As I'm responsible for the entire system, it's important to have a review of the overall picture, including all our hosts. To accomplish this, I have a separate activity configuration file and run Analog once a day with a **cron** entry. The activity configuration file includes the log files for all hosts, and this requires giving extra information to Analog so it can format the results; otherwise, / index.html would be considered as belonging to one host. Commands in configuration files must be on one line. The LOGFILE command allows you to specify the name of the host corresponding to the log file (ignore line wrap):

```
LOGFILE /var/log/httpd/access_log
 http://main-isp.com/LOGFILE /var/log/httpd/vhost1.com-access_log
 http://vhost1.com
```

Keeping Information Private

Our daily reports are published on the Web, so I prefer to keep cgi-bin information confidential. A daily webmaster e-mail report (described below) takes care of informing me of web-related exploits, so the information isn't required on the public reports. The cgi-bin directories and file names need to be aliased so that this information isn't available to the public. Analog can use output aliases to give control over how a file or directory is displayed within reports. This can be used to keep complete path and file names from the public, if desired. I use the following alias commands in my master.cfg files to translate cgi-bin path and file information to simply admin (ignore line wrap):

```
REQOUTPUTALIAS */cgi-bin/* "admin"
DIROUTPUTALIAS */cgi-bin/* "admin"
FAILOUTPUTALIAS */cgi-bin/* "admin"
FAILREFOUTPUTALIAS */cgi-bin/* "admin"
TYPEOUTPUTALIAS */cgi-bin/* "admin"
REFOUTPUTALIAS http://main-isp.com/cgi-bin/*  "admin"
REDIROUTPUTALIAS */cgi-bin/* "admin"
```

The last two items are used in Virtual Host master.cfg files, so we're still not giving away information on other local hosts in referral reports. If you want to be more specific, you could alias file names to match what they do, such as the following line (ignore wrap):

```
REQOUTPUTALIAS */cgi-bin/bannerpro.pl* "Banner Program"
```

Excluding Pages and Requests

I have a number of partial pages such as footers, sidebars and headers in a global directory that could cause Analog to inflate the request totals out of proportion. When you exclude information, it usually relates to the entire host, so it makes sense to use **exclude** commands in the master.cfg instead of in individual page or site configuration files. To exclude global directory accesses from being counted as requests, I use the command:

```
REQEXCLUDE */global/*
```

Partial web pages, such as header-and footer-type files, can also be excluded individually with the **PAGEEXCLUDE** command:

```
PAGEEXCLUDE */footer.html
```

or (for those who use PHP):

```
PAGEEXCLUDE */footer.php3
```

Daily Webmaster E-mail

I use a small script to receive a daily Webmaster report. This is basically the same as the Activity report, but it includes information that's excluded from the public version. When I read my e-mail in the morning, I can see the status of my system over the last 24 hours. The script runs from cron, and since Analog will send results to STDOUT if no outfile is listed, I use this to my advantage. The output becomes the body of the e-mail. A bare-basics webmaster.cfg file is included in Listing 5.

Listing 5

Automating with Scripts and cron

Using script files allows Analog to create several reports, one after the other. After the configuration files, the script files are the next step. I use one hourly script to create the Activity Report, and two daily script files: one for daily reports that are published on the Web, and the other to send the webmaster e-mail each morning. A monthly script file creates the text files that are mailed to customers each month. These scripts are shown in Listings 6-9.

Listing 6

Listing 7

Listing 8

Listing 9

Web Index for Published Reports

When working with the web-published statistics reports, or when simply checking them periodically, it's much easier if you create an index web page for reports and bookmark it. While this may seem a common-sense thing to do, I remember my early days of experimenting with Analog and fumbling around directories to find the resulting reports. Every step toward making analyzing our logs easier means we have a little extra time to work on developing other web projects to add to the list and continue the cycle.

Gaelyne Gasson (gaelyne@videocam.net.au) is the Web Administrator for VideoCam Services/VCSWEB, and the author of “The Internet for Commodore C64/128 Users”. Her interests include Commodore computers and all aspects of web activity. She's a proud member of the LinuxSA (South Australia) user group.

Archive Index Issue Table of Contents

Advanced search

Shell Functions and Path Variables, Part 2

Stephen Collyer

Issue #72, April 2000

Mr. Collyer continues his discussion with a detailed description of the addpath function.

In my previous article, I described a shell function that handled command-line options. In Part 2, we will use it in the path variable functions I promised to describe.

Each of the path variable shell functions is structured in a similar fashion. First, local variables are declared. Next comes the option-handling code, which employs the options function we became familiar with last month. Finally, the main functionality of the code is implemented. Because each function has the same structure, I will describe only one in detail this month. The next installment will describe various implementation features of the remaining functions.

We must first understand what the “environment” of a process is. The path variables we manipulate will usually be variables in the environment of a process, and we need our functions to alter their values (for example to add or remove directories).

In a nutshell, the environment of a process is a group of named variables (similar to shell variables) which are passed to any created child process. (A process, of course, is the entity which runs a program. If you type **ls** to a shell, for example, this creates a process to run the **ls** program.) A shell variable can be put into the environment by “exporting” it. For example, the commands

```
A=fred
export $A
```

create a shell variable called **A** and turn it into an environment variable. So, if you start a new process from this shell, it can examine its environment, find the **A** variable and notice it has a value of **fred**. Environment variables, therefore,

provide a one-way channel of information—from parent process to child. The parent and child processes don't share the environment variables—the child is given a new copy of them. Thus, if a child process changes the value of an environment variable, the parent will not be aware of the change.

Now, we *wish* to modify the path environment variables so that we can't start a new process when we run our shell utilities. Our utilities are implemented as functions, because functions run in the context of the calling process. Although they do not get a copy of the environment variables, they do have access to the existing set.

The addpath Function

The purpose of **addpath** is to add a pathel (path element) to a pathvar (path variable) in an idempotent fashion. Idempotent literally means “of equal power” and figuratively means “doing it *N* times is the same as doing it once”. So, for example,

```
NEWP=
addpath -p NEWP /abc
addpath -p NEWP /abc
```

adds **/abc** to the pathvar **NEWP** exactly once. **addpath** checks the pathvar to see if the pathel is already present. If not, it adds it; if so, it doesn't.

This function is helpful, because if you use it to add to your **PATH**, for example, you won't end up with multiple copies of the same directory in your path. The code to do this is shown split up into various listings to make discussion easier.

Listing 1

In Listing 1, we create some variables local to the function. The set in the first three lines is for options handling; the set in the final 4 lines contains variables specific to this function. The options handling variables tend to be very similar in each function that uses the options function.

Listing 2

In Listing 2, we handle the options supplied to the function. We do this by calling the options functions described last month. We tell it the names of the options we are prepared to handle, and give it a quoted list of the supplied arguments. When options returns, it will give us information on the supplied arguments in the form of variables which it has created. In addpath, we are prepared to handle **-h, -f, -b** and **-p**, (**-p** options. The -p option requires an argument, which is the name of a pathvar, such as **PATH**. When options returns, it also creates a variable called **options_shift_val**. We can use this to

shift away those command-line arguments it has already handled (i.e., arguments like **-h, -b** and so on). We do this immediately after the call to options. So, if the user had specified **-h** only, then **options_shift_val** would be set to 1, and we would shift away one argument; if **-b** and **-p** were specified, we would shift away three arguments (**-b**, **-p** and its required pathvar).

The next four "if" blocks appear in each pathvar function because they perform the following common tests:

1. If options created a variable called **opt_h**, then a **-h** argument was supplied and the user wanted some help. This we give by printing out usage information for the function and calling **return**. When a function call returns, it terminates, like a function call return in C. Don't make the mistake of calling **exit** in a function --this will terminate the shell process calling the function, which is probably not what you wanted to do.
2. We examine the **options_missing_arg variable**, which options creates if you didn't supply a required argument for an option. If this occurs, we print out the usage message, tell the user what went wrong and return.
3. We examine **options_unknown_option**. **options** sets this when you supply an argument that we're not prepared to handle (i.e., one that is not **-h**, **-f**, **-b** or **-p**, in this case). We return after giving the user some help.
4. Finally, we look at **options_num_args_left**. This gives us a count of the number of arguments that remain after we shifted away the ones that have already processed. The code here will be specific to each function, but for addpath, we require the user to specify the name of the directory to append. We check for this (sloppily) by complaining if no arguments are left at this point.

So far, we have performed the type of options processing that crops up over and over in shell scripts and functions that take arguments. We have checked that we haven't been supplied any options we don't know about, and that required arguments have been given. We have also provided a basic help facility via the **-h** handling.

Listing 3

Listing 3 shows the option handling code specific to the addpath function. The first two lines in this section set values for the variables **COMMAND** and **pathvar** to be used later. These lines may seem a little cryptic at first sight, but essentially they set up the default path variable to which we add (**PATH**) and the default command we execute to add to it. Note that there are single quotes around the contents of **COMMAND**. The shell will store the literal string between the quotes into **COMMAND**, and no variable substitution will be attempted.

If we type **addpath /def**, then pathvar will contain **PATH**, **sep** will contain **:** and **dirname**, will contain **/def**. Later, when we evaluate a line of code containing **COMMAND**, the shell will substitute the literal strings **${pathvar}**, **${sep}** and **${dirname}** with their values to produce "${PATH}:/def". Note that we used three individual characters inside **COMMAND**: **$**, **{** and **}**. this ensures the shell interprets them literally and that they appear in the evaluated output.

The next two lines of code override the defaults if either the **-f** or **-p** option is given. If **-f** was given, the user wishes to add to the front of the pathvar and we set **COMMAND** accordingly, putting the pathvar variable at the end. If **-p** was given, the user supplied the name of a path variable; the options function stores this in **opt_p** and we set the **pathvar** variable to this value. So, if the user typed:

addpath -f -p NEWP /def

then **opt_p** and thus **pathvar** will contain **NEWP**, and **COMMAND** will look like **$dirname$sep\$$pathvar**, with the **dirname** variable sitting at the front.

Next, we set the value of the **sep** variable. Usually we want this to contain **:**, as that character separates path elements. However, if the path variable to which we are adding is initially empty, we don't want anything in **sep**. This ensures we don't add leading or trailing **:** characters to the path. The three lines of code starting **sep=:** implement this.

Finally, we store the name of the path element to be added in **dirname**. Note that at this point, it will be in **$1**; any other command-line arguments were shifted away immediately after options returned.

Listing 4

In Listing 4, we actually add the path element to the path variable. The first thing to note is that the real work of the function takes five lines of code. We have already performed all the required setup and argument checking, so there is little left to do. Essentially, we check to see whether the path element is already present in **pathvar**; if not, we add it.

```
  element=$(eval echo \$$pathvar | colon2line |
    grep -x "$dirname")
```

First, note the **variable=$(...)** syntax. This is equivalent to the older **variable="..."** syntax and means "Run the commands within the brackets, and use what they write to standard output as the value of variable." This is called command substitution. In our example, we have a pipeline of commands; the output of the final command (**grep**) becomes the value of element. Let's look at each

command in the pipeline. Remember that **$pathvar** contains the name of the path variable we wish to add to, and **$dirname** contains the name of the directory to add.

Assuming the value of **$pathvar** is **PATH**, the command line is **eval echo pathvar** first expanded by the shell to **eval echo $PATH**. Next, because of the **eval**, the shell reevaluates the line. This time it, expands **$PATH** into something like "/usr/bin:/bin:/usr/bin/X11" and echoes its contents into the next command in the pipeline.

colon2line is another shell function, with code we have not yet seen. It merely prints each element of a colon-separated string on a separate line (i.e., it converts the **:** character to a newline character). This can be done in many ways. Here, for example, is an **awk** one-liner for this purpose:

```
awk `BEGIN{RS=":"}{print}'
```

This command tells awk to assume that **:** separates records in a piece of text, and then to print each colon-separated record it sees. Each separate line is then read by the third command in the pipeline.

The command **grep -x "$dirname"** is used to check for the presence of the path element we wish to add. The shell will replace **$dirname** with its value before grep is executed. It is surrounded by quotes so we can correctly handle the pathological case of **$dirname** containing spaces.

We tell grep we want only exact matches (**-x**). This ensures that if we run the following commands:

```
addpath /abc
addpath /ab
```

we add both of the distinct path elements. Without -x, grep would report it had seen a match when the second command was run, because "/ab" is a substring of "/abc".

If grep sees **$dirname** in its input, it writes the name to the standard output. Because we are using grep in command substitution brackets, its output is assigned as the value of the element variable.

In a nutshell, the pipeline ensures the element has a non-null value if it is already present in the path variable, and a null value otherwise. If the element is null (**["$element" = ""]**), we add it with the following command:

```
eval eval $pathvar=$COMMAND
```

By now, you should be familiar with the purpose of the eval command: it tells the shell to reevaluate the line being processed, and on each evaluation, shell variables are expanded. The only question here is, why do we need two of them? Assume the user types the following:

```
addpath /abc
```

Assume further that PATH contained /usr/bin only, and that /abc was not already present in $PATH. Let us look at how this line is expanded step by step.

1. $pathvar=$COMMAND: initially
2. PATH=\$\{${pathvar}\}${sep}${dirname}: after first shell expansion
3. PATH=${PATH}:/abc: after first eval
4. PATH=/usr/bin:/abc: after second eval

It should be clear that if we had applied only one eval to the command, the shell would have executed the command in step 3 and we would have replaced the current contents of **PATH** with the literal characters "${PATH}:/abc". The final **eval** forces the replacement of the embedded "${PATH}" string with its current value. Remember also that the **PATH** variable we alter here is not local to this function; any changes made are visible to the calling shell.

That ends the description of the addpath function. The structure of the other pathvar functions is very similar to that of addpath, so in Part 3, I will describe only one or two points of interest about each of them.

Stephen Collyer (stephen@twocats.demon.co.uk) is a freelance software developer working in the UK. His interests include scripting languages and distributed and thread-based systems. Occasionally, he finds the time to talk to his wife and two remarkably attractive and highly intelligent children.

Archive Index Issue Table of Contents

Advanced search

The Generation Gap

Brian Marshall

Issue #72, April 2000

An examination of the issues involved with the use of open-source software components in closed-source applications.

Use of open-source software is mushrooming with the use of Linux. The benefits of open source are becoming more widely known—software quality is improved from peer-review, and future risk is reduced if the source is available.

Open-source projects depend on contributions by motivated, talented developers. Big, important projects like Linux attract many contributors; a project for a software development tool may interest a variety of software developers; UNIX systems gurus contribute to projects for Internet tools. However, a project for software that identifies butterflies may not attract much interest, and the only people who are going to contribute to a project for a company's accounts receivable system are those who are paid to do so. Not all software benefits equally from being open source.

The development of closed-source software will continue. As Eric Raymond observes in “The Magic Cauldron”, some applications (as opposed to operating systems and utilities) can be good candidates for being closed-source. The owner of an application may find more benefit in the software being secret and/or rentable than being open source, at least for a while.

It may be undesirable for the source of a homegrown business system to be open, because it reveals so much about the business's procedures and capabilities. A company may want to keep the source of its business systems confidential.

The owner of an application may want to keep the source closed so that the software can be rented (i.e., licensed for money). Many applications are developed only when someone believes they can make money renting them.

While the benefits of open source are becoming more widely appreciated, closed-source software will continue to be developed.

Open-Source Components in Closed-Source Applications

Closed-source applications can benefit from using open-source components.

Applications generally make use of pre-existing software components. Almost all programs use general-purpose function libraries that come with compilers. Many applications provide a front end using a GUI framework that comes with the compiler or a third party. Some software uses special-purpose libraries to do things like statistical analysis or mapping projections. Many applications include, or know where to find, a self-contained “component” that provides sophisticated functionality, such as a database engine or a map-drawing system.

There is less risk in using a third-party component if it is open source. This is particularly true of components developed by small companies and individual developers. It's very difficult to develop software components and make money renting them. People think, “I don't know these guys. Who knows what they've done in there? My application depends on this functionality.” They also think, “What if these guys disappear? Or find something else to do? Or change the component so I can no longer use it? Or not fix what I need fixed?” They may also think, “This component does what I need to do now. What about next year? What if I want some enhancements? Are these guys going to care what I want?” Being open source addresses these concerns. Developers can see what an open-source component does and how it does it. They always have the option of making improvements themselves (or paying someone to do it).

If a component is open source, its owner cannot rent it. However, money can be made in support services, and the people best poised to do so are developers. A tiny outfit without the credibility to rent a component may find that the best way to make money is to make it open source.

It might be easier to attract contributors to an open-source project for a component than to a project for a narrow application. The users of a component are programmers, and they make up a natural pool of potential contributors whereas the users of a narrow application might be, for example, dentists.

Developers may be motivated to improve the component if an improvement is all that is necessary for the component to be used. Companies using the component may be motivated to spend money to improve it.

Objections to Collaborating With the Enemy

Concerns arise over using open-source components in closed-source applications. The most commonly used open-source license is the General Public License (GPL). It grants users the freedom to modify the software and distribute this derived work only if, among other restrictions, the derived work is also licensed under the terms of the GPL. This license ensures that all users of a piece of software, even users of modified versions, have the freedom to see and modify the source code. The Free Software Foundation promotes the use of the GPL.

A component licensed under the terms of the GPL cannot be used in a closed-source application, but some developers of open-source components would like to see them used in any application, open-source or not. The Library General Public License (LGPL, also known as the Lesser General Public License) was developed for this situation. A component licensed under the LGPL may be included in a closed-source application as long as users can get the source for the component, modify it and rebuild the application.

In many cases, however, developers of closed-source applications cannot (or will not) make use of components licensed under the LGPL. Developers of commercial products and companies writing software for their own use frequently don't want any given component badly enough to submit to the restrictions of this license.

There are less restrictive open-source licenses. A component licensed under the terms of the MIT License can be used by anybody for anything. Such licensing makes a component practical to use in closed-source development. Many people in the Open Source community find this license to be subversive, precisely because it makes it practical to bring open-source software into closed-source applications.

Open Source and the Hacker Culture

The people who write open-source software are the hackers—the community that had its origins at the MIT AI Lab.

It began in 1959, when a group of people in the Signals and Power group of the Tech Model Railway Club got access to a small computer—the TX-0. This was different from the big IBM mainframe, where one submitted a deck of punched cards, then waited around for their own job to run. With the TX-0, one could actually sit at a console and deal with the machine directly. The club loved it. They loved the challenge of trying to make it do what they wanted, and the feeling of power when it finally did. They started with practically nothing in the

way of an operating system. They wrote their own system functions and simple tools and used them to build better tools.

These were the original hackers—a group of people who thought programming was the most important thing in the world. They were explorers. They were a community. They shared information and ideas, and they shared the belief that if information and ideas move freely, everyone benefits. They shared the product of their work—tools, utilities, experiments, games and jokes. A person's standing in the hacker community was based on almost nothing except that work.

People moved in and out of the community at MIT. Some hackers moved to other institutions and kept in touch via the ARPA-Net (a precursor to the modern Internet). The community became a network community and stopped having a specific geographic location. As Internet access became available to anyone in college, this community grew. The Internet became the focus for many hackers; its infrastructure was developed largely by hackers. Now the focus has shifted to Linux, and the software that runs under it.

As Eric Raymond describes in his paper “Homesteading the Noosphere”, the hacker culture is a gift culture—status within the community is based on what a person creates and gives away. Open-source software is the product of people programming (testing and debugging) and giving away the results. Most of this work is done in the spare time by people who also hold paying jobs.

People contribute to open-source projects for a variety of reasons: fascination with the problem, desire for a meaningful programming project, wanting to participate in the community, an opportunity to prove themselves, or a desire to use the new software personally. They continue to contribute to the community and culture because it works. People who have never physically met, each doing whatever they feel like doing, can in their spare time create great software.

The General Public License is seen as strongly supporting the hacker culture. It can be difficult to attract contributors to a project for software that is not licensed under the GPL. It can be particularly difficult for projects involving components licensed under the MIT license.

Components and the Open Source Movement

In the past, people in the Open Source community were the primary users of open-source software, but this has changed. Many new Linux users are not programmers and many of them will never contribute to open-source projects. Open-source developers don't feel cheated by this; they feel vindicated. To a

hacker, the ultimate measure of software's worth is whether people find it useful.

Linux has been very good for the Open Source movement. Huge numbers of people are trying it. Many more are hearing about it. People are pushing for its use where they work. More and more people are learning about the Open Source movement and its values, and becoming involved in open-source projects and a part of the Open Source community. The community benefits from increasing use of open-source software, whether that use directly supports open-source development or not. The community benefits when dentists, for instance, use Linux.

Suppose, however, that a dentist and I want to get together and write an application that implements his top-secret super-duper billing technique. The technique is a trade secret, so the application must be closed source. Now, suppose that as I'm writing it, I want to use an open-source library to do statistics. If this makes my billing application a “derived work” and I'm going to get a bunch of licensing hassles, I'll do something else. I'm not trying to derive a new statistics library—I just want to use it. If it's a good thing for a dentist to run his billing application on Linux, why wouldn't it be a good thing for the application to use open-source parts?

The more people who use open-source components, the more people will decide some component needs a wee change and they should do a little work for the open-source project. If open-source components are used in business applications, businesses will, from time to time, find it in their interest to pay someone to work on the components. This is good for the Open Source movement.

Generations of the Internet

The Internet has gone from being tiny to small to huge to gigantic. The use of open-source software has gone from being tiny to small to huge. The parallels between open source and the Internet are particularly interesting, because the cultures overlap—the infrastructure of the Internet was developed by and is maintained by hackers.

The Internet and its culture can be viewed as having passed through four generations:

1. The U.S. Defense Department connects the ARPA-net with other defense networks to support military research. This proves to be a great way for researchers to communicate and share information. Demand for networking grows, but access is limited to a small community of computer

science researchers, government employees and government contractors. A culture based on sharing is established.

2. The National Science Foundation commissions the NSF-NET to connect colleges with five supercomputer centers. The NSF pays for campus connections if access will be generally available to students. Anyone at a four-year college can get on the network. Everything still revolves around sharing; commercial use is forbidden by the people providing the funding.
3. Demand for networking increases, and commercial Internet Service Providers appear. Commercial use of the Internet is still frowned upon (and sometimes flamed upon), but for people and businesses paying for their own access, there is no one to forbid it. The emphasis shifts from sharing to using—surfing, chatting, e-mailing, browsing, finding cool stuff and being entertained. Commercial use increases. The idea of a business having a web site goes from being flaky to being obvious. The original culture, with its strict mores enforcing an ethic of sharing, is apparently losing its dominance.
4. Millions and millions of people are on the Internet. You can do just about anything you want. Strict mores have given way to true freedom. It's changing the world. More people are sharing on the Internet than ever before. It's the hacker spirit on a gigantic scale.

Conclusion—Closed Source as Users of Open Source

Linux began as an operating system used primarily by programmers. Now it is used by all kinds of people, and the proportion of Linux users who are programmers is steadily decreasing. Most of these non-programmers know very little about the traditions of the hacker culture. Many of them use Linux simply to make money, and it's just about the best thing that could have happened to the Open Source movement.

Open-source software isn't going to replace closed-source software any time soon. People write closed-source applications because they believe they can make money renting them, and they will continue to believe this. Companies write software for their own use, and want to guard their trade secrets.

Closed-source development is a wonderful market for open-source components. The more companies use open-source components, the more they will contribute to open-source development.

It's great that programmers can create something like Linux in their spare time, but they also spend a lot of time at their paying jobs. Component development may be where the Open Source movement invades company time.

Resources

Acknowledgements

Brian has been a software developer in Calgary's oil and gas industry since 1981. He is deeply into C++ and object-oriented design. His career began vowing never to learn Cobol; it progressed to never learning VB and now involves never learning MFC. Brian first got into UNIX in 1991 but he has only been using Linux for about six months. **Brian Marshall** can be reached via e-mail at bmarshal@agt.net.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Enlightenment Basics

Michael Hammel

Issue #72, April 2000

A guide to getting and installing Enlightenment.

Enlightenment consists of a core package, a set of libraries upon which it is dependent and add-on packages known as “epplets”, which are not required to get E (Enlightenment) running. You can grab the core package and the libraries it requires from the primary E web site at http://www.enlightenment.org/. At a minimum, you'll need the packages listed in Table 1 in order to run on a Red Hat 5.2 system. Red Hat 6.0 or later, SuSE 6.2 and later and the current Debian distributions already have E available. If you install everything, you'll have E ready to run. If not, you will need to determine which package includes E. Usually this is the same category in which you find GNOME or KDE when choosing packages to install.

Table 1

The graphics libraries you'll want are TIFF, GIF, PNG and JPEG, but most of these have been installed with Linux distributions for a couple of years, so chances are you already have them. Like the other libraries upon which E is dependent, you can find the source distributions for the graphics libraries on the E web site.

The first thing you want to do is make sure the graphics libraries are present. The quick way to check for them is to type **ldconfig -p**. This command will print all libraries initially installed on your system. If you add more libraries in other directories (as we'll be doing in a moment) and have configured them in /etc/ld.so.conf, it will show those as well.

```
ldconfig -p | grep -i gif
ldconfig -p | grep -i jpeg
ldconfig -p | grep -i tiff
ldconfig -p | grep -i png
```

If any of these commands simply return without printing anything, you're missing the associated library. The **grep** command will allow only the output from **ldconfig** which contains the string that follows it (**-i** means ignore case). If you're missing a graphics library, get it from the web site and install it. Since most people will already have these, we're going to skip their installation.

Next, you want to install imlib. This library handles the display of graphic images (backgrounds, borders, etc.). It scales images appropriately, such as when you make a window wider. Imlib is not installed on most older distributions of Linux, and some newer versions may not install it by default, so we'll look at the installation of this library directory from its source.

Grab the library source from the E web site and unpack it into a local directory. In this directory, run the configure script as follows:

```
./configure --prefix=/usr/local/imlib
```

This will prepare the package for compiling. The “--prefix” option tells configure that, after compilation, installation will be into a directory called /usr/local/imlib. Management of packages like imlib, that are built from source rather than an RPM, is easier if you install them in their own directories.

After running the configure command, you're ready to build the software by typing **make**. This will run for a time and should complete without error; however, you may see some warnings go by. On my system, I don't have the GIF library installed, so I get a message stating “Native GIF support will not be built.” This isn't a big problem unless you plan on using GIF images in your personal themes or as background images. The themes provided in the E core distribution all use PNG, so at a minimum you'll need the PNG library installed.

You might also run into a warning about GTK not being found and one that reads “gdk_imlib will not be built”. This shouldn't happen to you unless, like me, you have GTK installed in its own directory. Most readers will have installed GTK during installation of the operating system. If you do get this error, either you may need to upgrade your GTK package or you need not have it installed in a standard location. In either case, you don't have to have gdk_imlib in order to use the imlib library with E.

After the source code build completes, you're ready to install it. Since we specified a directory in /usr/local, we'll need to be root to install the software there. You can either log out and log back in as root, or just run the **su** command to change to the root user. The latter is easier to do, but you will need the root password. As root, type

```
make install
```

Now, edit the /etc/ld.so.conf file and append the following line to it:

```
/usr/local/imlib/lib
```

Then run **ldconfig**. This will tell the system new libraries are installed under /usr/local/imlib/lib and to check for libraries there when running programs or compiling them. We'll be doing this step again for the other libraries, so be sure you remember it. If you used su to change to the root user, type **exit** to return to your normal user name. Remember: you should never work as the root user *except* to manage system files. You can compile these packages without being root, and you should—it's safer that way.

Now you're ready to install the font libraries, fnlib and freetype. Fnlib, a collection of fonts that E uses, has some dependencies on imlib, so that's why imlib had to be installed first. Freetype is the library which gives E internal support for TrueType fonts. In this way, E can use TrueType fonts even if your X server doesn't support them.

Unpack fnlib and the FreeType library into their own separate directories. In the fnlib directory, run this command:

```
./configure --prefix=/usr/local/fnlib \
 with-imlib-prefix=/usr/local/imlib
```

In the FreeType directory, run this:

```
./configure --prefix=/usr/local/freetype
```

In each directory, run **make**, change to the root user again and run **make install**. Edit the /etc/ld.so.conf file and append these two lines:

```
/usr/local/fnlib/lib
/usr/local/freetype/lib
```

Finally, rerun ldconfig and exit from the root user.

Normally, this would be all that was necessary for other programs which use **auto-conf** (the tool which creates the configure scripts) to find your newly installed packages. But E is missing a minor bit in its configure scripts, so we have one extra step to do. We need to make symbolic links for the files in the fnlib and freetype directories into their respective directories under /usr/local. It's not hard, so don't let this technical explanation scare you. Just type the script in Listing 1 into a file called /tmp/fixit.sh, then type:

```
sh /tmp/fixit.sh
```

Listing 1

That should do it. Seems like a lot of work, and it is, but this is what you have to live with when working with very young software. Linux may be in its infancy, but E is only a bit older than most zygotes.

After installing the libraries upon which E is dependent, we're ready to build and install E itself. This one follows the same basic steps as the others—run **configure**, **run make**, run **make install**. Except this time, you don't need to update the ld.so.conf file.

```
./configure --prefix=/usr/local/enlightenment\
 with-imlib-prefix=/usr/local/imblib
```

Change to the root user and type **make install**, then exit from the root user.

When you run the configure script, you will get a notice about Esound not being found. That's fine—you don't need it to work with E. As far as I'm concerned, sound on a computer does one of two things: makes noises when you type, or the clock changes and plays MP3s on your CD. Playing music shouldn't be the job of the window manager, so the latter option isn't important. As for making sounds when you type, well, any window manager that does that should be placed on a floppy disk and nailed to the stake where its author is tied just before they light the fire at his feet. But I digress.

The process for building E is just like building the libraries, so the installation should go smoothly. Chances are, the only real problems you'll hit might be if the libraries weren't installed correctly. If that's the case, reread this section and try again. After the build and installation are complete, you may want to add the Enlightenment/bin directory to your path:

```
export \
PATH=$PATH:/usr/local/enlightenment/enlightenment/bin
```

or link the binaries to /usr/local/bin:

```
ln -s \
/usr/local/enlightenment/enlightenment/bin/enlightenment\
/usr/local/bin
```

Note that E's main program—the window manager—is a program called “enlightenment”.

What looks like a typo is not—the path is correct in both of these commands. We told the build process to install all the enlightenment tools under /usr/local/enlightenment. When we ran **make install**, the E installation process created a number of directories under /usr/local/enlightenment. One of these was called enlightenment. The reasons for this are technical, but suffice it to say the Enlightenment developers had one installation scenario in mind and that

differs from mine. It doesn't matter. It still works using my method, and later upgrades to E will be easier to handle without disturbing any other packages.

Michael J. Hammel (mjhammel@graphics-muse.org) is a graphic artist wanna-be, a writer and a software developer. He wanders the planet aimlessly in search of adventure, quiet beaches and an escape from the computers that dominate his life.

Archive Index Issue Table of Contents

Advanced search