Advanced search

Linux Journal Issue #94/February 2002

Ultimate SUBVERSION: Building a Better CVS
LINUX JOURNAL
The Real Stars
Webcam Tools
NLE Video Editors
SOHO
Need a SUPERIOR small or home office work space?
Printing at REMOTE LOCATIONS painlessly
Setting up a home WIRELESS NETWORK

Using ssh Port Forwarding to Print at Remote Locations

Rory Krause

Issue #94, February 2002

Rory shows you how to connect the printing systems on different networks across the Internet in a secure manner.

In 1997, I was talking to a professional investor who thought that the value of office space in urban high rises was about to take a big dive. His rationale was that telecommuting was going to get *big* real soon. Well, here we are five years down the road, and I don't think telecommuting really has exploded. I've done it a couple of times. Some people here at SSC telecommute a couple of days a week. But in downtown Seattle, high rises are going up all over the place. On any given day there are 10-20 construction cranes on the skyline.

I wonder if my professional investor friend also was selling interests in pulp and paper companies. Tomorrow's office will be paperless, right?

Whether you telecommute or not, you probably use some form of electronic connectivity to the office when you are not there. Can you check your work e-mail at home? Do machines send e-mail to your cell phone? Can you access your company's intranet from the Web? Or, maybe you have never even been to the corporate headquarters, and you rove the globe packing a laptop that connects to your company's headquarters via a virtual private network (VPN).

If you don't have your VPN set up yet by your IT staff with its unlimited budget, you might be interested in ssh.

ssh--oh yeah, that's a secure Telnet program, right? Yes, it is, and it's much, much more. You're not still using Telnet, are you? Previous issues of *Linux Journal* have talked about the “much, much more” of ssh (see Resources). In this article I show a specific example of using ssh to do “much, much more”. I also demonstrate how to use ssh's port-forwarding feature to connect the printing systems on different networks across the Internet, at the same time securing the data while in transit.

Imagine that you work at Example Company. Example Company has a traditional office with a network of computers that contain all the data and programs that you and the company use to produce widgets. You are at home and use ssh to connect to Example's network over the hostile-worm-infested-jungle known as the Internet. **ssh** allows you to log in, read mail and start X11 programs. You also can use scp to pass files back and forth securely. If you have a fast connection to the Internet and through to your company's network, things can be very similar to working at your desk inside the traditional office—up until the time you want to burn a hard copy. What can you do? Well you can print to a file, scp the file to your home computer and submit the job to the local printer with lpr. This method works, but it takes three steps and does not work if you cannot print to file.

Hmmm...well you could turn all three steps into one big command-line juggernaut. On your home computer, typing

```
    ssh -f rory@example.com cat secretdata | lpr
```

fires up an ssh session over to example.com and concatenates the file called secretdata in your remote home directory. The f option makes ssh go into the background after password authentication but before the cat command is run. Piping to lpr takes the standard out of ssh and directs it to standard in for lpr on the local computer. As long as your home lpr system is working and knows how to handle the file format of secretdata, out pops a hard copy on your home printer. Now, where did you put that ream of paper?

It would be nice just to be working away in your shell at Example Company and type **lpr secretdata**. That would save 30 typed characters and be easier to remember. What about printing from a mail reader or printing a report from a database application? An integrated print system would be even more useful for these activities.

One way to make the printing system seamless is to set up a relay system that gets the print job from the lpr system at example.com and passes it over to the lpr system at home.

Let's do this by using the port-forwarding features of ssh. What this does is allow you to connect an arbitrary port on your local system to a port on the remote system or vice versa. Then you can use programs that talk to a port to connect to the remote machine. Let's look at an example. Say you wanted to run your browser on your local machine but to look at your company's intranet, which is hosted on the company server at example.com but is not available to the public. Using ssh you forward the local port 8080 to example.com's port 80 with the command:

```
ssh -L 8080:example.com:80 example.com
```

After authentication you get a shell on example.com. Then, on your home computer open up your favorite browser, and enter the URL http://localhost:8080/.

Your browser sends its requests to the local port 8080, which gets passed over to port 80 at example.com, which processes the request and sends the answer back over the same channel to be displayed in your browser. Similarly, you could do a MySQL query or talk to an SMTP server by forwarding a local port to the appropriate port on a remote host.

Now that we have a handle on forwarding ports, let's talk about printing. First, we need a relay program that runs between the two different lpr systems.

Let's use a client/server system. The idea is that when we ssh to example.com, we fire up a local client program that connects to a server over at example.com using a forwarded port. If the server has a print job waiting, it sends it back to the client, which passes it off to the local lpr.

Examples of a simple server and client are shown in Listing 1 and Listing 2 [Listing 2 is available at ftp.linuxjournal.com/pub/lj/listings/issue94/5462.tgz].

Listing 1. rlserver

The server will be started by lpr on the remote system and will run only if there is a print job waiting to be sent from the remote machine over to the home machine. The remote lpr system host is set up to pass the print job off to the rlserver program using this entry in /etc/printcap:

```
# Remote Printer for Rory
rory
        :lp=|/usr/local/scripts/rlserver 8888
        :sd=/var/spool/lpd/rory
        :lf=/var/spool/lpd/rory/log
        :mx#0
        :sh
```

Printcap entries vary slightly on lpr systems; this one is for LPRng. The 8888 is an argument to rlserver telling it which port should be used for Rory. Other users can be set up with different ports—just make sure the ports aren't in use by some other program. Ports 1-1024 usually require special privileges.

What the client does is poll the server once every five seconds. It connects to the local port that has been forwarded. If there is a job waiting, then there is a server to connect to. Otherwise the connection is refused and the client tries again in five seconds. When a connection is made, the server sends the print job data over to the client, which copies it down into a temporary file. After the client receives all the data, the temporary file is sent to the local lpr. Now there are a few other things we need to do to set this up to run smoothly.

If you actually work in the corporate office part of the time, you want to use the office printer while you are in the office. But when you log in from home, you want the lpr-forwarding printer to be selected. This can be set up automatically by using an ssh feature and the shell initialization files. In $HOME/.ssh/enviroment, add the line:

```
REMOTE=yes
```

Once you ssh to a remote host, this variable will be set in your remote environment.

Next, add this stanza to your .bashrc on the remote host:

```
if [ "$REMOTE" = "yes" ]; then
    PRINTER=printername
fi
```

This will set your printer to *printername* when ssh starts a bash session. Make sure printername is the same as what is set up in the printcap file. If you don't use bash, do whatever your shell needs in its startup file.

Next, let's make sure the port forwarding takes place behind the scenes. That way we can simply type **ssh example.com** without remembering a lot of options and port numbers. Edit the $HOME/.ssh/config file, and add

```
Host example.com
    LocalForward 8888 example.com:8888
```

This will forward the local port 8888 to example.com's port 8888.

If you have a slow internet connection you may want to bump up the compression level from the default of six. Add these two lines before the host-specific sections of $HOME/.ssh/config:

```
Compression yes
CompressionLevel 9
```

Copy the program rlclient (Listing 2) to your home computer. Make sure the first line has the correct Python path in it. Execute rlclient in a shell. In a separate shell ssh to example.com. Your port 8888 should be forwarded, compression turned on, and your printer variable all should be set up. Check the printer variable with:

```
echo $PRINTER
```

It should say whatever you set up as your printername.

Now make sure your local (home) lpr system is up and running, and send a file to print on the remote system (example.com):

```
lpr mytestfile.txt
```

Check on the status of the print job with **lpq**. Also, watch the shell where rlclient is running. It will display output of what is going on.

I had some problems with new versions of ssh that use Protocol 2 not forwarding ports or complaining each time rlclient tried to connect. The error messages were of this sort:

```
channel_request_forwarding: cannot listen to port: 8888
channel_open_failure: 3: reason 2 Connection refused
```

In these cases, it was necessary to force the protocol level to 1 by using **-1** on the command line or adding Protocol 1 to the $HOME/.ssh/config file under the other entries for example.com.

In addition to the lpr client and server, I found two very simple programs useful in debugging port forwarding. These are called time.client.py and time.server.py (Listings 3 and 4, respectively). Both programs take one argument, which is the port number you want them to use. The server waits for connections, sends out the time of day when it receives a connection, and the client simply prints out the time. They can be used over ssh or on the same machine.

Listing 3. time.client.py

Listing 4. time.server.py

Once everything is debugged, it is much easier to use a script to start the rlclient in the background when sshing to example.com. Then use the script (or an alias to the script) to ssh:

```
#!/bin/sh
# Start up the rlclient and ssh to example.com.
$HOME/rlpr/rlclient > $HOME/rlpr/log &
echo "$!" > $HOME/rlpr/pid
ssh example.com
kill -9 $(cat $HOME/rlpr/pid)
```

In this example, we have looked at how to relay print jobs from one location to another over a port secured by ssh. Uses for this port-forwarding feature are limited only by your imagination. Go forth and ssh.

Resources

email: info@linuxjournal.com

Rory Krause is a system administrator at Specialized Systems Consultants. He holds a degree in Mechanical Engineering from the University of Washington. Hobbies include archery and gardening.

Archive Index Issue Table of Contents

Advanced search

Setting up an All-Linux Wireless LAN

Don Marti

Issue #94, February 2002

Laptops are cheap, WiFi cards are cheap and well-supported...maybe it's time to build your own LAN.

If you've been to a Linux conference lately, you've probably seen a lot of people with wireless network cards, happily checking their e-mail while they sip refreshing beverages. The 802.11b, or “WiFi”, cards are really cheap, convenient and well supported under Linux.

If you want to do wireless networking at home, though, just plugging in a base station might not be too smart. Even with the largest key sizes, the badly named Wired Equivalent Privacy (WEP) encryption protocol that WiFi uses is vulnerable to people walking nearby—for large values of “nearby”. Security consultant Peter Shipley reports connecting to WiFi networks 25 miles away using a directional antenna. And as little as half an hour of intercepted traffic could be enough time for someone to break WEP and get on the Net using your base station (see Resources).

You can do some things to make WEP stronger, but it's inadequate as protection. (Adobe Systems, Inc. reportedly shut down all their WiFi base stations after they got the government to arrest programmer Dmitry Sklyarov. They may be trying to set software and free speech back 500 years, but they aren't stupid.)

The threat to WiFi is not that people might read your data—you're using ssh and SSL to protect sensitive information anyway. The threat is people using the network you're responsible for to do bad things. If a spammer or a script kiddie gets on the Net using your base station, it's almost impossible to find him or her. Who gets the blame? You do.

If you do decide to provide public connectivity via 802.11b, that's mighty neighborly of you. Get a separate net connection, put up a base station, join the relevant mailing lists and have fun. If someone uses your public base station to send spam or break into people's computers, and your ISP cancels the account, you'll still have your regular net connection.

So if off-the-shelf base stations are bad, what's the alternative? How about an old laptop? In case you haven't noticed, prices are coming down, and you probably can get a new laptop with half a gig of RAM and a 1600 × 1200 display for about eight dollars by the time you read this. So treat yourself to a new laptop and convert your old one into the base station. The on-line auction sites also have made damaged laptops really cheap. Busted hinge? Missing battery latch? Might not be the best thing to carry around, but with a little duct tape, there's your base station.

I put Debian on mine because it has packages for all the software mentioned in this article and a good automatic upgrade system. But any current distribution will work. (I didn't have to compile anything from source or reboot to do this article, although I did reboot a couple times after configuration to make sure everything came up in the right order.)

I shouldn't have to say this, but buy WiFi cards that are well supported by your distribution of choice. I got a pair of Orinoco cards.

Since your base station will be exposed to anyone who walks by with a WiFi card, take a few basic administration and security measures (which apply to any internet server) before you plug the card in.

First, forward root's mail to your real e-mail address, so important messages don't go unread.

Check that log rotation is working and that plenty of space is available in /var/log.

Configure the mailer dæmon to listen only on the loopback interface.

Set up either ntpd or a cron job to run ntpdate, so all times in the logs will be correct.

Remove all unused software, and apply any security fixes or updates.

Copy your main ssh key into .ssh/authorized_keys2 in your home directory on the base station, so that you never need to send your password over the Net, even encrypted. Configure sshd to refuse passworded or root logins.

Make sure you are subscribed to the security mailing list for the distribution the base station is running.

Finally, run Nmap against the base station from outside to make sure that no unnecessary services are running. From outside you should see ssh and that's it.

Just in case your base station does get compromised, make sure that no information or keys on the base station would help people get into any of your other systems. For example, accounts on the base station should not have ssh keys.

In this article, I use 10.2 addresses for the underlying WiFi, and 10.3 addresses for the VPN. Now it's time to get on the air.

Set up the PCMCIA Cards in the Base Station

The actual order of eth0, eth1 numbering depends on the order in which the drivers are insmod-ed, and the cardmgr dæmon probes the slots in order on boot. On my laptop (now base station) the WiFi card and conventional Ethernet card will only fit if the WiFi card is in the top slot, slot 0. That means that my WiFi interface is eth0, and my regular Ethernet is eth1.

By default, when you boot the base station, the PCMCIA script will start the cardmgr dæmon and possibly exit before all the cards are initialized. If you're running a server of any kind, this is not what you want to happen. All the interfaces should be up before dæmons try to start. Pass the -f option to cardmgr by putting it in the PCMCIA init script or configuration file; on Debian it's /etc/pcmcia.conf (see Listing 1). Below is /etc/pcmcia/wireless.opts:

```
# use "Ad-Hoc" mode to act as a base station.
# Set your own ESSID.
case "$ADDRESS" in
*,*,*,*)
    ESSID="wifi.ssc.com"
    MODE="Ad-Hoc"
    ;;
esac
```

Listing 1. /etc/pcmcia/network.opts

To check your work, do an **ifconfig** and an **iwconfig** on the base station, and make sure the information is correct. Make sure that you can still log in to the base station over the regular Ethernet.

Set up WiFi on the Client

All you should have to do is put in an extended service set ID (ESSID) that matches the one on the base station.

Set up the DHCP Server on the Base Station

To make the base station work as a DHCP server, you also will need to add a route to 255.255.255.255, which is the destination address for DHCP traffic. Unless the 255.255.255.255 route exists, DHCP packets will take the default route instead of the WiFi interface, which is not what you want. You can add this route to the dhcp init script. While you're editing this script, make dhcpd run it only on the WiFi interface (in the case of solanum, eth0). You don't want the base station spewing DHCP traffic to places it isn't wanted. So replace

```
/usr/sbin/dhcpd
```

with

```
route add -host 255.255.255.255 dev eth0
/usr/sbin/dhcpd eth0
```

Set up a dhcpd.conf file on the base station to give out IP addresses only to your own systems:

```
# /etc/dhcpd.conf for solanum
# run the DHCP server on the WiFi interface only!
default-lease-time 1800;
max-lease-time 7200;
subnet 10.2.0.0 netmask 255.255.0.0 {
}
subnet 198.144.202.0 netmask 255.255.255.0 {
}
host cannabis {
    hardware ethernet 00:02:2d:2e:56:df;
    fixed-address 10.2.0.2;
}
```

This is not a security measure, but it will keep your DHCP server from wasting time on any of your neighbors who set up their clients incorrectly.

At this point you should be able to ping the base station from the client over WiFi.

VPN

There are many virtual private network (VPN) options for Linux, and you might have one installed already for a different reason. If so, you can skip installing a separate VPN just for WiFi, and simply configure IP masquerading on the base station to allow traffic only from the WiFi network to the VPN server and vice versa, and you're done. If you need to set up a VPN between several locations for travelers or for home offices, and you also need VPNs for WiFi at each location, save time by picking one VPN that works for both.

Otherwise, choose and install a VPN just for WiFi. For this article, I chose vpnd (see Resources), which has the advantages of working with an “out-of-the-box” kernel, being available as a Debian package and being simple to configure.

The kernels of clients and the server will need to have the kernel random number generator and SLIP support as a module. The stock kernels that come with distributions have this, but if you built your own kernel and didn't compile any modules you didn't need then, you'll have to go back, **make menuconfig**, choose SLIP and then do:

```
make modules && make modules_install.
```

The good news is that you don't have to reboot to do this if you're running a modular kernel and still have the kernel source and kernel .config that you built from. If you took out kernel random number generator support, shame on you—put it back in, as not only vpnd but much other fine crypto software depends on it.

To set up keys for vpnd, run **vpnd -m** on the base station, then copy the resulting /etc/vpnd/vpnd.key to the client. Configuration files for vpnd are pretty simple; Listing 2 shows an example.

Listing 2. /etc/vpnd/vpnd.conf

At this point you should be able to ping the base station's virtual address (10.3.0.1 in this case) from the client, and vice versa. If not, check the logs for vpnd errors, and use **ifconfig** and **route** at both ends to make sure the IP address and routing information are correct.

IP Masquerading

Every distribution has its own IP masquerading setup tool, and IP masquerading articles are as common as pig tracks, so turn it on however it says in the book. You will need to make sure that the WiFi network (10.2.0.0/16 in our example) doesn't get masqueraded—just the VPN. To test that masquerading is set up correctly, don't only surf the Web and send mail from the client—temporarily change the default route on the client to go over the WiFi directly instead of the VPN, and make sure that you can't.

Many exciting future developments in wireless networking are on the way. Future security protocols should make the VPN dance unnecessary, and community networks such as NoCatNet are working out protocols to let you share your access point with neighbors without opening yourself up to abuse. But, today's 802.11b cards are going to be common and serviceable for a long time.

Resources

email: dmarti@zgp.org

Don Marti is technical editor of *Linux Journal* and editor in chief of *Embedded Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

The Subversion Project: Buiding a Better CVS

Ben Collins-Sussman

Issue #94, February 2002

This open-source project aims to produce a compelling replacement for the Concurrent Versions System (CVS).

If you work on any kind of open-source project, you've probably worked with CVS. You probably remember the first time you learned to do an anonymous checkout of a source tree over the Net, your first commit or learning how to look at CVS diffs. And then the fateful day came: you asked your friend how to rename a file. “You can't”, was the reply.

“What? What do you mean?” you asked.

“Well, you can delete the file from the repository and then re-add it under a new name.”

“Yes, but then nobody would know it had been renamed.”

“Let's call the CVS administrator. She can hand-edit the repository's RCS files for us and possibly make things work.”

“What?”

“And by the way, don't try to delete a directory either.”

You rolled your eyes and groaned. How could such simple tasks be so difficult?

The Legacy of CVS

No doubt about it, CVS has evolved into the standard software configuration management (SCM) system of the Open Source community, and rightly so. CVS is free software and has a wonderful nonlocking development model that allows hundreds of far-flung programmers to collaborate. In fact, one might argue that, without CVS, it's doubtful whether sites like Freshmeat or SourceForge ever would have flourished as they do now. CVS and its semi-chaotic development model have become an essential part of Open Source culture.

So what's wrong with CVS? Because it uses the RCS storage system under the hood, CVS can only track file contents, not tree structures. As a result, the user has no way to copy, move or rename items without losing history. Tree rearrangements are always ugly server-side tweaks.

The RCS back end cannot store binary files efficiently, and branching and tagging operations can become very slow. CVS also uses the network inefficiently; many users are annoyed by long waits, because file differences are sent in only one direction (from server to client, but not from client to server), and binary files are always transmitted in their entirety.

From a developer's standpoint, the CVS codebase is the result of layers upon layers of historical "hacks". (Remember that CVS began life as a collection of shell scripts to drive RCS.) This makes the code difficult to understand, maintain or extend. For example, CVS's networking ability was essentially stapled on. It was never designed to be a native client/server system.

Rectifying CVS's problems is a huge task, and we've only listed a few of the many common complaints here.

Enter Subversion

In 1995, Karl Fogel and Jim Blandy founded Cyclic Software, a company for commercially supporting and improving CVS. Cyclic made the first public release of a network-enabled CVS (contributed by Cygnus software). In 1999, Karl Fogel published a book about CVS and the open-source development model it enables (cvsbook.red-bean.com). Karl and Jim had long talked about writing a replacement for CVS; Jim even had drafted a new, theoretical repository design. Finally, in February 2000, Brian Behlendorf of Collabnet (www.collab.net) offered Karl a full-time job to write a CVS replacement. Karl gathered a team and work began in May.

The team settled on a few simple goals: it was decided that Subversion would be designed as a functional replacement for CVS. It would do everything that CVS does, preserving the same development model while fixing the flaws in CVS's (lack of) design. Existing CVS users would be the target audience; any CVS user should be able to start using Subversion with little effort. Any other bonus features were decided to be of secondary importance (at least before a 1.0 release).

At the time of this writing, the original team has been coding for a little over a year, and we have a number of excellent volunteer contributors. (Subversion, like CVS, is an open-source project.)

Subversion's Features

Here's a quick rundown of some of the reasons you should be excited about Subversion:

- Real copies and renames: the Subversion repository doesn't use RCS files at all; instead, it implements a virtual versioned filesystem that tracks tree structures over time (described below). Files *and* directories are versioned. At last there are real client-side mv and cp commands that behave just as you think.
- Atomic commits: a commit either goes into the repository completely or not all.
- Advanced network layer: the Subversion network server is Apache, and client and server speak WebDAV(2) to each other. (See the “Subversion Design” section below.)
- Faster network access: a binary diffing algorithm is used to store and transmit deltas in *both* directions, regardless of whether a file is of text or binary type.
- Filesystem properties: each file or directory has an invisible hash table attached. You can invent and store any arbitrary key/value pairs you wish: owner, perms, icons, app-owner, MIME type, personal notes, etc. This is a general-purpose feature for users. Properties are versioned, just like file contents. And some properties are auto-detected, like the MIME type of a file (no more remembering to use the -kb switch).
- Extensible and hackable: Subversion has no historical baggage; it was designed and implemented as a collection of shared C libraries with well defined APIs. This makes Subversion extremely maintainable and usable by other applications and languages.
- Easy migration: the Subversion command-line client is very similar to CVS; the development model is the same, so CVS users should have little trouble making the switch. Development of a cvs2svn repository converter is in progress.
- It's free: Subversion is released under an Apache/BSD-style, open-source license.

Subversion's Design

Subversion has a modular design; it's implemented as a collection of C libraries. Each layer has a well defined purpose and interface. In general, code flow begins at the top of the diagram and flows downward—each layer provides an interface to the layer above it (see Figure 1). Let's take a short tour of these layers, starting at the bottom.

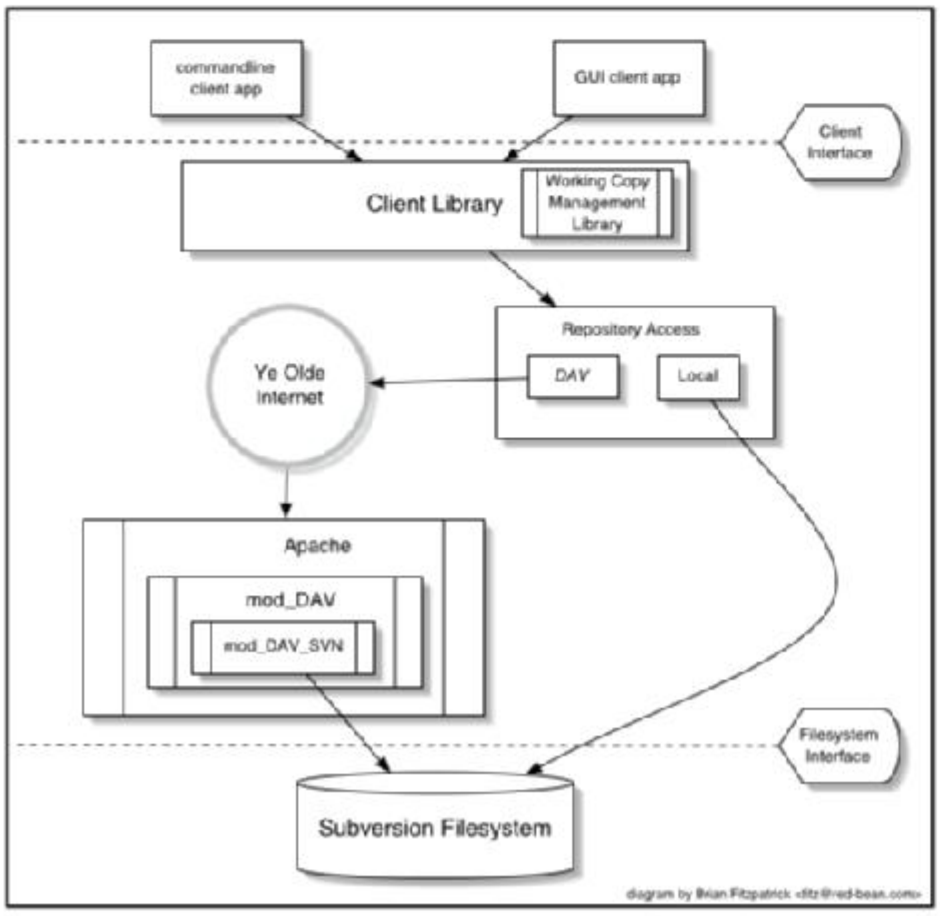


Figure 1. Design Layers of Subversion

The Subversion Filesystem

The Subversion Filesystem is not a kernel-level filesystem that one would install in an operating system (like the Linux ext2 fs). Instead, it refers to the design of Subversion's repository. The repository is built on top of a database, currently Berkeley DB, and thus is a collection of .db files. However, a library accesses these files and exports a C API that simulates a filesystem, specifically a versioned filesystem.

This means that writing a program to access the repository is like writing against other filesystem APIs: you can open files and directories for reading and writing as usual. The main difference is that this particular filesystem never loses data when written to; old versions of files and directories always are saved as historical artifacts.

Whereas CVS's back end (RCS) stores revision numbers on a per-file basis, Subversion numbers entire trees. Each atomic commit to the repository creates a completely new filesystem tree and is individually labeled with a single, global revision number. Files and directories that have changed are rewritten (and older versions are backed up and stored as differences against the latest version), while unchanged entries are pointed to via a shared-storage mechanism. This is how the repository is able to version tree structures, not just file contents.

Finally, it should be mentioned that using a database like Berkeley DB immediately provides other nice features that Subversion needs: data integrity, atomic writes, recoverability and hot backups. See www.sleepycat.com for more information.

The Network Layer

Subversion has the mark of Apache all over it. At its very core, the client uses the Apache Portable Runtime (APR) library. In fact, this means that a Subversion client should compile and run anywhere Apache httpd does. Right now, this list includes all flavors of UNIX, Win32, BeOS, OS/2, Mac OS X and possibly NetWare.

However, Subversion depends on more than just APR; the Subversion server is Apache httpd itself. Why was Apache chosen? Ultimately, the decision was about not re-inventing the wheel. Apache is a time-tested, open-source server process ready for serious use, yet is still extensible. It can sustain a high-network load. It runs on many platforms and can operate through firewalls. It's able to use a number of different authentication protocols. It can do network pipelining and caching. By using Apache as a server, Subversion gets all these features for free. Why start from scratch?

Subversion uses WebDAV as its network protocol. DAV (distributed authoring and versioning) is a whole discussion in itself (www.webdav.org), but in short, it's an extension to HTTP that allows reads/writes and versioning of files over the Web. The Subversion Project is hoping to ride a slowly rising tide of support for this protocol; all of the latest file browsers for Win32, Mac OS and GNOME speak this protocol already. Interoperability will (hopefully) become more and more of a bonus over time.

For users who simply wish to access Subversion repositories on local disk, the client can do this too; no network is required. The Repository Access (RA) layer is an abstract API implemented by both the DAV and local-access RA libraries.

This is a specific benefit of writing a “librarized” revision control system; it's a big win over CVS, which has two very different, difficult-to-maintain code paths for local vs. network repository-access. Feel like writing a new network protocol for Subversion? Just write a new library that implements the RA API.

The Client Libraries

On the client side, the Subversion working copy library maintains administrative information within special /SVN subdirectories, similar in purpose to the /CVS administrative directories found in CVS working copies.

A glance inside the typical /SVN directory turns up a bit more than usual, however. The entries file contains XML that describes the current state of the working copy directory (and that basically serves the purposes of CVS's Entries, Root and Repository files combined). But, other items present (and not found in /CVS) include storage locations for the versioned properties (the metadata mentioned in the “Subversion Features” section above) and private caches of pristine versions of each file. This latter feature provides the ability to report local modifications and do reversions without network access. Authentication data also is stored within /SVN, rather than in a single .cvspass-like file.

The Subversion client library has the broadest responsibility. Its job is to mingle the functionality of the working-copy library with that of the repository-access library, and then to provide a highest-level API to any application that wishes to perform general revision control actions.

For example, the C routine svn_client_checkout() takes a URL as an argument. It passes this URL to the repository-access library and opens an authenticated session with a particular repository. It then asks the repository for a certain tree and sends this tree into the working-copy library, which then writes a full working copy to disk (/SVN directories and all).

The client library is designed to be used by any application. While the Subversion source code includes a standard command-line client, it should be easy to write any number of GUI clients on top of the client library. Hopefully, these GUIs should someday prove to be much better than the current crop of CVS GUI applications, which are no more than fragile wrappers around the CVS command-line client.

In addition, proper SWIG bindings (www.swig.org) should make the Subversion API available for any number of languages: Java, Perl, Python, Guile and so on. In order to Subvert CVS, it helps to be ubiquitous.

Subversion's Future

The release of Subversion 1.0 currently is planned for the beginning of 2002. After the release of 1.0, Subversion is slated for additions such as i18n support, intelligent merging, better changeset manipulation, client-side plugins and improved features for server administration. Also on the wish list is an eclectic collection of ideas such as distributed, replicating repositories.

A final thought from Subversion's FAQ: “We aren't (yet) attempting to break new ground in SCM systems, nor are we attempting to imitate all the best features of every SCM system out there. We're trying to replace CVS.”

If in three years Subversion is widely presumed to be the standard SCM system in the Open Source community, then the project will have succeeded. But the future is still hazy. Ultimately, Subversion will have to win this position on its own technical merits. Patches are welcome.

email: sussman@red-bean.com

Ben Collins-Sussman has worked for 11 years as a programmer and system administrator at various government, academic and commercial institutions. Ben currently works for Collabnet, Subversion's main sponsor, and also moonlights as a composer in the Chicago theater community. His home page is at www.red-bean.com/sussman.

Archive Index Issue Table of Contents

Advanced search

3-D Programming with Python

Jason Petrone

Issue #94, February 2002

Jason scratches the surface of OpenGL programming techniques using PyOpen GL, a suite of Python modules.

Graphics programming can be tedious. Linking against large 3-D libraties increases compilation time. Because a lot of fine tuning is often necessary for everything to look perfect, stretches of minor changes buried between long builds are commonly encountered. These lengthy debug cycles make 3-D graphics an ideal application for prototyping in a high-level language like Python.

Extensions to a number of 3-D graphics APIs are available for Python. For IRIX systems, the Python distribution comes with a module providing access to the SGI IRIS GL library. Python programs can make use of the Java3D API from inside JPython, an implementation of Python that runs inside a Java Virtual Machine. This article focuses on the OpenGL library because of its widespread use and excellent support for Linux and Python.

Downloading and Installing PyOpenGL

PyOpenGL is a suite of Python modules that provides access to OpenGL as well as an assortment of helper utilities and extensions to complement OpenGL's low-level interface. It was originally created by James Hugunin, Thomas Schwaller and David Ascher. Tarn Burton recently has taken over as lead developer, and Rene Liebscher and Michael Fletcher also maintain the package.

Since OpenGL wrappers make up the bulk of PyOpenGL's functionality, you will need a basic understanding of OpenGL to write programs with it. There are many excellent tutorials and references available on OpenGL, see Resources for a list of recommendations.

The first requirement for PyOpenGL is OpenGL itself. If you don't have an OpenGL implementation installed already, check your GNU/Linux distribution to see if it includes the packages, or download the Mesa 3-D graphics library from www.mesa.org. For PyOpenGL to work at full capacity, the module Numerical Python must be installed. Sources for Numeric and PyOpenGL can be found at numpy.sourceforge.net and pyopengl.sourceforge.net, respectively. Compilation and installation is easy thanks to Greg Ward's distutils module, which is included in Python as of version 1.6. Running the command **python setup.py install** from inside the unpacked source directories should build and install the modules. Before installing from source, you may want to check if your GNU/Linux distribution already provides these modules. They were included as part of my Debian distribution. Note: the version I worked with is PyOpenGL 1.5.7, since the time of this writing, version 2.0 has become available.

A Simple Python OpenGL Application

The OpenGL specification does not define specifications for interaction with windowing systems. Consequently, programs using OpenGL must use an external GUI toolkit. The program in Listing 1 uses GLUT, a cross-platform windowing toolkit for OpenGL. Unless you are using a commercial OpenGL implementation, you probably already have GLUT installed.

Listing 1. Opening a Window, Setting up Lighting and Drawing a Teapot with GLUT

This code opens a window, sets up lighting and draws a teapot. Aside from the added syntactic compactness Python affords, it looks much like the equivalent program written in C. One minor difference is how the display function callback is set. Setting the display function callback in C or C++ would only require calling the function glutDisplayFunc(display). Setting the callback in PyOpenGL is done in two steps: invoking glutSetDisplayFunc() and then glutDisplayFunc(). This idiosyncrasy also applies for setting other callbacks such as glutMouseFunc() and glutReshapeFunc().

While GLUT is suitable for most small OpenGL applications, it still requires a fair amount of work to implement functionality that is often desirable when testing, such as mouse control for zooming, panning and rotation. Togl is a Tkinter widget that automatically provides these features as well as default lighting. Listing 2 shows the same program using Togl.

Listing 2. Same Program Using Togl

Notice it uses considerably less code but provides much more functionality. The cost of this is flexibility. If Togl's default lighting and user interface don't meet your requirements, you will need to re-implement them yourself. Togl is excellent for prototyping, as it eliminates the need to write and debug boilerplate lighting and navigation code.

PyOpenGL also integrates well with other GUI toolkits that have 3-D widgets. Bindings exist for wxWindows, FLTK, FOX and GTK.

Using Texture Maps

Texturing mapping is the technique of taking image data, like a photograph, and "gluing" it to a polygon. Using textures requires converting the external texture images into one of the low-level internal formats supported by OpenGL. While the code required for setting textures in Python does not differ much from normal OpenGL code, Python does simplify the process of loading and manipulating texture image files.

The Python standard library includes rgbimg, a module for reading SGI imglib(.rgb) files. While this may seem like an obscure format, its simplicity allows its inclusion in Python's standard library without causing too much bloat. Using the GIMP or ImageMagick, you can convert images from formats like PNG, JPEG or TIFF to SGI imglib. The command **rgbimg.longimagedata(*`file`*)** will read and decode the specified SGI imglib formatted file and return it as a binary string of four-byte RGBA pixels. This data can be passed to OpenGL functions like glTexImage2D, using the format parameter GL_RGBA and the data type GL_UNSIGNED_BYTE.

Another useful module in the Python standard library is imageop. This module includes functions for cropping, scaling and grayscale conversions for images. The imageop functions operate on data in the same GL_RGBA/ GL_UNSIGNED_BYTE format as the data returned from rgbimg.longimagedata().

PyOpenGL Utility Functions

In addition to wrapping the core libraries and toolkits, PyOpenGL also includes a number of utility functions to perform high-level tasks. Have you ever wanted to grab a still image of an OpenGL scene? You always can use programs that support screen capture, such as xv or The GIMP, but what if you want to grab a still at a specific point in time? Taking screenshots manually won't work if the program needs to run without user control, like a CGI script. You also can't grab a dozen stills a second manually to create a movie. PyOpenGL has a couple helper functions for automatically grabbing scene shots in various formats.

Listing 3 is an adaptation of the previous program that saves a PostScript image of the scene to disk, then exits. You can take a look at the image that was saved in Figure 1.

Listing 3. Saving a PostScript Image, Then Exiting

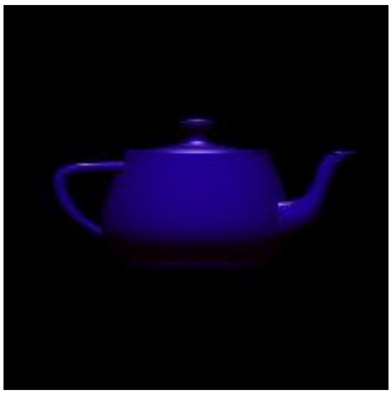


Figure 1. Results from Listing 3

The scene is saved using the command

```
    openglutil.glSaveEPS('ex3.eps', 240, 240)
```

The first argument is the name of the file to save to. The second is the width in pixels to capture, and the third is the height. Using the same syntax with the glSavePPM() command will save an image of the scene in the portable pixmap file format. Additionally, if you have compiled PyOpenGL with TIFF support, the command glSaveTIFF() is available for saving scene shots in the TIFF format.

PyOpenGL also provides a convenience wrapper for OpenGL's object selection or “picking” mechanism. The selection mode of operation in OpenGL automatically tells you which objects fall within a given region of the view screen. This means you can pass the selection mechanism to a point on the screen where a mouse click occurred, and it will tell you what object was clicked on.

Here are the basic steps for setting up the selection mechanism in PyOpenGL:

1. Write a method or function that draws each object in the scene wrapped in between a call to glPushName() and glPopName(). Pass a different integer as an argument to glPushName() for each independent object in the scene. OpenGL will use this value to indicate which objects were selected.
2. Set up the event handler for mouse clicks. The way this works depends on the toolkit you are using. In Tkinter, mouse-click event handlers are set up by passing a method or function reference to the bind() method.
3. Once a mouse click has occurred, pass the mouse x and y coordinates, along with a reference to the method or function created in part one, to the function OpenGL.GL.glSelectWithCallback(). This function will return a list of tuples of the form (near, far, names) where near and far represent integer depth values, and names is a tuple of the names passed in

glPushName() for the selected objects. If no objects were selected, an empty tuple is returned.

Listing 4 demonstrates use of the selection mechanism in PyOpenGL. This demo draws a cube and a sphere on the screen. Experiment with holding down the Ctrl key and pressing the left mouse button with the pointer in different places.

Listing 4. The Selection Mechanism in PyOpenGL

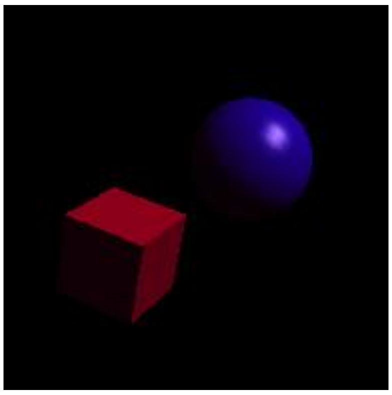
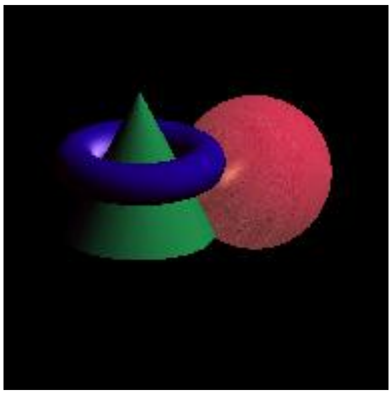
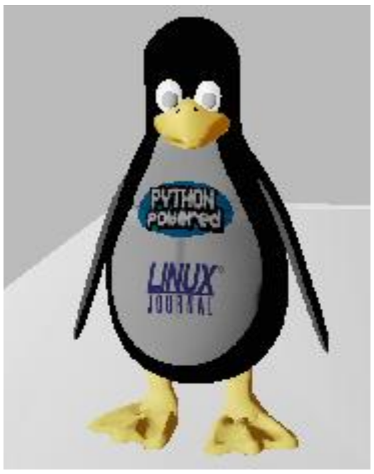


Figure 2. Results from Listing 4

The fog.py Demo Included with PyOpenGL


A Simple Scene with Lighting and Textures Rendered Using PyOpenGL



A 3DStudio Model Rendered Using PyOpenGL

The selection mechanism is the simplest way to perform this task, but it is by no means the only way. Another technique is to render the objects to a back buffer that OpenGL does not send to an output device. To do this, render each object using a different color. Once done, the back buffer should contain the 2-D rendering of the scene. By examining the color at a particular point you can determine which object was drawn in that region of the screen. This back buffer is called the feedback buffer in OpenGL.

Yet another method for object picking is to do the intersection testing by hand. This means you would have to project a ray from the view screen and test each object in the scene for intersection. Once you have found a list of intersected objects, you then must sort by depth values to see which was closest to the view screen. While this may offer more control over the process, it is an arduous task and is difficult to do efficiently in Python.

Improving Performance

Now that you have seen how to write simple OpenGL programs, you are probably wondering if Python can scale up to the demands of more advanced 3-D applications. While the performance of a PyOpenGL program generally lags behind that of its C or C++ counterpart, optimization techniques can narrow the gap considerably.

The main strategy in improving execution speed is to reduce the amount of time spent within the Python interpreter by moving expensive operations into native code. One means of accomplishing this is to rewrite sluggish parts of the program in a fast, natively compiled language like C or C++. Implementing these compiled portions of the program as Python extension modules allows the remaining interpreted Python code to access their functionality. While this approach certainly has potential for speeding things up, it lacks the simplicity of a pure Python solution. It also requires you understand how to write Python extension modules in a language that compiles to native code. Besides, if you wanted to do it in C, you wouldn't have started messing around with Python in the first place!

OpenGL display lists provide a way to move operations into native code without any of the headaches associated with the former approach of writing extension modules. Display lists allow OpenGL programs to cache a set of commands further down in the rendering pipeline. In some environments, OpenGL even can store display lists on the graphics card itself, far away from the bottleneck of the Python interpreter.

The glGenLists() command creates an array of empty display lists. It takes a single integer argument, the number of display lists, to generate. It returns the number of lists that were successfully created. Wrapping a set of OpenGL operations with the commands glNewList() and glEndList() fills a specified display list. Once stored, subsequent invocations of that set of operations requires only a single command, glCallList(). The syntax for using display lists in PyOpenGL is pretty much the same as in OpenGL with C.

Next Steps

We have just begun to scratch the surface of OpenGL programming techniques in Python. For more information, make sure to check out the documentation that comes with PyOpenGL or on-line at pyopengl.sourceforge.net/documentation/index.html.

Resources

Jason Petrone (jpetrone@acm.org) is a member of the technical staff at the Corporation for National Research Initiatives (CNRI) in Reston, Virginia. He first discovered the powerful combination of OpenGL and Python while working on a CAVE virtual reality project at his former place of employment, the National Center for Supercomputing Applications (NCSA).

Using Debian Apt-get over Freenet

Timm Murray

Issue #94, February 2002

Although its reputation suggests the contrary Freenet does have many mirroring uses.

Freenet has generated a lot of hype for being the “son of Napster” or some equally degrading report. For Freenet to survive, it needs to get away from being just another way to get illegal MP3s. In legal terms, Freenet needs “massive non-infringing use”. Sorry, but getting the latest copy of *Quake* doesn't count as non-infringing.

This article describes a use for Freenet that few will find a problem with (unless you use an RPM-based distro, to which I say “Get a real package manager!”). It also once again illustrates the power of small, flexible tools that do one thing and do it well.

I Want My .debs!

As a Debian user who often uses my distribution of choice in production environments, I am confronted with problems in choosing mirror FTP sites from which to download new packages. Sometimes a mirror site is already overloaded with users, sometimes it isn't holding the packages I need and still other times it fails completely. I could resort to using the official ftp.debian.org, but that's not very polite. It would be nice if I didn't have to think about any of this.

Fortunately, the creators of Freenet have given me an answer and didn't even realize it at first. While Freenet has much to do with anonymity, it also provides a highly efficient mirroring system that no centralized system could hope to achieve.

The system described in this article requires no new code for either Freenet or Apt-get. It employs a standard user interface add-in for Freenet called FProxy, which originally was intended for the user to access Freenet through a web browser. Since it communicates with the browser via HTTP, and Apt-get already understands how to talk HTTP, this process only requires running a few simple commands and modifying the /etc/apt/sources.list.

Before proceeding, be sure to download the latest version of Freenet (0.3.9.1 as of this writing, although it will probably change by the time you read this) from www.freenetproject.org. Set up a node on your local machine and use some test files to insert and request. Open your favorite browser and point it to http://localhost:8081. If it brings up a page for requesting and inserting files, everything should be set up correctly. Be warned that it may take awhile for FProxy to come up, depending on the speed of your machine and which Java Virtual Machine you're running.

On a side note, you can get a Debian package that installs Freenet for you. It is available either at the web site above or in Debian unstable. Unfortunately, due to a buggy implementation of the big number classes, Freenet cannot work on the current version of Kaffe (1.0.6), and thus it can only run on a non-Free virtual machine; you'll find it in “contrib”. If you have a problem with this, you either can use a patched version of Kaffe, the latest CVS version of Kaffe or wait until the next stable version arrives. Current plans are to make future releases of Freenet compliable under GCJ, which brings Java code into native machine code, bypassing the problem of a non-Free VM completely.

In case you were wondering, Freenet makes rather heavy use of strong cryptography, so you'll find it in the non-US area, too. Edit your sources.list as needed.

What to Do

FProxy was meant to be the default user interface for Freenet. Although it is not a full proxy server, it communicates to the browser through HTTP (this is what was shown earlier when you opened up your browser to http://localhost:8081). After realizing that Apt-get could already understand HTTP, it was seen that getting packages from Freenet would be quite simple.

Before going any further, make a backup of your current /etc/apt/sources.list. I didn't do this the first time, and it took a few annoying hours to get that system to download packages correctly again.

Now, add this line to the real sources.list:

```
deb http://localhost:8081/debian-test/dists stable
```

For more information, see the sources.list(5) man page. You'll need to modify http://localhost:8081 to reflect the port on which FProxy is running. This port is set in the configuration file .freenetrc, located in the top-level directory in which the Freenet tarball was decompressed. If you used the Debian package, it will be in /etc/freenet/freenetrc.

You also need to add application/x-debian-package to FProxy's passthroughMimeTypes. This will be in fproxyrc (prior to 0.3.8.1) or you can find it in freenetrc under services.fproxy.passthroughMimeTypes.

Next, go to a Debian mirror and download (from the directory /debian/dists/stable/main/binary-<*your-arch*>/, replacing <*your-arch*> as necessary) Packages, Packages.gz and Release. Insert all of these (see below). Now download the “hello” package from the devel directory on the Debian mirror and insert that too, being sure to place it under exactly the same name. Now you should be able to run **apt-get update** (which will download the Packages file off your node) and then **apt-get install hello**, which will download and install the GNU Hello program. If it worked, congratulations! You can now get packages off Freenet.

To insert these packages, first you will need to create an SVK key pair. This allows you to insert into a subspace that only someone with the private key can access. To do this, enter

```
  freenet_insert -makeKeypair &> keypair.svk
```

Now, open up keypair.svk in your favorite text editor and delete everything except the value of the private key, and save the result into prv.svk. You now can insert with:

```
  freenet_insert -serverAddress 127.0.0.1:<port>
  SSK@`cat prv.svk`/<dir>/<filename>
  <filename>
```

replacing <*port*> with the port number your Freenet node is running on, <*dir*> with the “directory” (Freenet doesn't really have directories, but we can pretend it does) that you want to insert under and <*filename*> with the name of the file. You can retrieve the document by going back into keypair.svk, deleting everything except the value of the public key and saving it under pub.svk. Then type

```
  freenet_insert -serverAddress 127.0.0.1:<port>
  SSK@`cat
  pub.svk`/<dir><filename> <filename>
```

replacing things as described above. You can allow others to access these files by sending them the public key.

This method avoids a problem that arises with a simpler, guessable key type called KSK, which is actually a subspace where the private key is publicly known. The problem is, there is no way of knowing if a KSK actually has the data it's supposed to. Thus, it can be replaced by an attacker inserting a different document at a hops-to-live value of 1 and cutting the link before the transmission to the next node is done. Doing this enough times will replace the document completely for the majority of Freenet. This means that it would be easy to replace, say, the libc6 package with a virus. Think about how packages generally have to be installed from root, and you see what kind of damage this could do. Even with a subspace, it is strongly suggested that you check the MD5 sums of the packages you're downloading.

The Everything Over Freenet (EOF) Project keeps a list of currently maintained distributions that are under subspaces, which cannot be taken over without knowing the private key associated. Subspaces are slightly more complicated to use but infinitely safer. Please see eof.sourceforge.net/APT for instructions on this.

What Else Is There?

So you can get Debian packages off Freenet. Big deal. You could have done this with the FTP or HTTP servers you had been using all along. Not only that, but now you have to worry about packages dropping out of the store just because they aren't popular enough.

Well, it is a big deal. The huge list of mirrors that are a pain to keep maintained are now largely unnecessary, replaced by automatic updating within Freenet. This relieves a lot of bandwidth congestion.

Still, how do we keep packages from falling out? On a healthy Freenet, a file would have to be very unpopular to ever fall out. Needless to say, the current Freenet isn't so healthy (although having a few thousand *über*-stable Debian boxen might help). However, we need a way to ensure that little-known but oh-so-needed packages will remain available.

One suggestion is to create a wrapper script around Apt-get. This will first invoke **apt-get install <*package*>** through a Freenet request. If the package is unavailable, it will automatically download it from an FTP or HTTP mirror, e-mail the Freenet maintainer for your particular Debian distribution (provided you have mail set up properly on your machine) and warn them the package has dropped out of Freenet. Then it actually will install the package it downloaded.

But that won't work. You would have to know the private key of the subspace for it to be of any use to anybody except you. On the other hand, documents in a subspace are almost always going to be redirects to a CHK (a key where the name is a hash of the data, so the same data will result in the same CHK name). If the redirect still exists but the data under the CHK has fallen out, you could re-insert the CHK. However, the best we can hope for if the redirect document fell out is to e-mail the maintainer to ask them to re-insert.

Moving along to other thoughts, we come to the problem of the unstable distribution, which provides a unique challenge to Freenet. Current versions provide no real means to update content. This does not work well with the fast-moving Debian unstable. A hack of using date- and version-based redirects actually was put into Freenet in response to the problem of keeping Debian unstable the same in Freenet and FTP servers. Even so, this still takes a lot of bandwidth to accomplish.

The concept of date-based updating (putting the date in the name of the key and re-inserting every day) was brought up, specifically due to the question of how to update Debian packages. My personal feelings are that date-based updating is fine for relatively small web sites, but it's too much for the number of files being put into an entire Debian distro. Others may disagree.

In any case, those FTP or HTTP mirrors still will be needed to some extent. However, most of the mirrors are now unnecessary as most of the bandwidth can be distributed efficiently by Freenet.

Also note that many new means to make Freenet more efficient are being placed in the upcoming 0.4 version of Freenet. The 0.3 version was broken in some fundamental ways, making only the most popular files easily available. With any luck, we should see more files survive in version 0.4 of the network, which may very well be available by the time you read this.

Help Us Mirror

One great way to help the Freenet and Debian projects is to help us mirror the Debian packages in Freenet. Scripts are available to aid in this; we just need people with a lot of bandwidth to help.

Basic requirements for helping the mirroring effort are that you have a Freenet node running and are willing to use a lot of bandwidth all at once. These are preferable so your ISP won't come knocking because you blew up a T3 trying to insert all this stuff. This isn't something to do casually; the current Debian 2.2r2 main archive for i386 weighs in around 2GB. The non-US version is quite a bit smaller, so if you don't live in the US and don't have much bandwidth, this might be a good option for you.

Complete information on helping mirror Debian packages can be found on eof.sourceforge.net/APT. There you will find the script for mirroring and a list of what distributions need to be mirrored.

Freenet Is for Music Pirates and Pornographers

Freenet can do much, much more than provide an efficient way of distributing porn; it is also an efficient way of distributing the latest 'N Sync album. More than that, though, this article has shown that Freenet does have far greater uses.

email: hardburn@runbox.com

Timm Murray is a student who spends his free time trying to come up with a useful contribution for Freenet.

Configuring pppd in Linux, Part I

Tony Mobily

Issue #94, February 2002

Connecting to the Internet may be easier than you think; Tony begins this two-part series with how to configure your modem.

Take Command

Configuring pppd in Linux, Part I

Connecting to the Internet may be easier than you think; Tony begins this two-part series with how to configure your modem.

by Tony Mobily

Today, many people install Linux on their desktop computers. Unfortunately, quite a few people seem to get stuck as soon as they try to do the one thing that apparently no one is ready to give up: connecting to the Internet. Why do they get stuck? There are several reasons, but the main one is that there is no official equivalent of Windows' “remote access” or Mac's ConfigPPP. As a result, many users end up having some similar programs available on their system (GNOME dial-up, Linuxconf, KDE dialer, etc.) without knowing anything about them.

It is also quite possible that no automatic configuration program is available (for example, if the user chose not to install either GNOME nor KDE when he or she installed the system). So why not take on the challenge of learning how to establish an internet connection by hand? This article explains how to set up a modem on your Linux system—a crucial step that many users seem to have trouble with. In the next article I will talk about how to configure pppd itself, assuming that the modem is configured correctly.

What You Need to Know

To get the most out of this article you should first be fairly familiar with the basics of using the shell (how to change directory, list the files in a directory, etc.), editing a file using any editor, running a program and using virtual consoles or several terminals in the X Window System.

This article will assume that you have a real modem, not a Winmodem. Configuring a Winmodem is possible but can be tedious and is outside the scope of this article.

Finding the Modem

The first thing you have to do is know where your modem is. What we are looking for is the serial port to which the modem is connected. This is true even if you are using an internal modem, as the modem card itself will contain a serial port. Your computer is likely to have two serial ports. Quite possibly, one is being used by your mouse (unless you have a PS/2 or USB mouse). In UNIX, every device is represented as a file in the directory /dev. This directory contains an entry for every device you possibly could have installed on your computer. The serial ports are called ttyS, followed by a number between 0 and 3. Let's have a look:

```
cd dev
ls -l ttyS*
crw-------  1 merc   tty    4,  64 Aug  3 10:24 ttyS0
crwxr-xr-x  1 root   tty    4,  65 Aug  3 10:25 ttyS1
crw-------  1 root   tty    4,  66 May  6  1998 ttyS2
crw-------  1 root   tty    4,  67 May  6  1998 ttyS3
```

Which is the one the modem is connected to? The answer is, it depends on where you plugged in your modem (or, if you have an internal modem, it depends on how it is configured). If you are used to calling serial ports COM1, COM2 and so on, you should know that the equivalents here are COM1 = ttyS0, COM2 = ttyS1, COM3 = ttyS2 and COM4 = ttyS3. And, if you don't know which serial port your modem is plugged into (or configured for) don't worry, because we will find that out in a little while.

Usually, there is a symbolic link (that is more or less the equivalent of a Windows shortcut) called “modem” that points to one of the serial ports. For example, on my system I have:

```
ls -l modem
lrwxrwxrwx  1 root  root  5 Feb  7  2000 modem -> ttyS1
```

In my case the modem is connected to the second serial port, ttyS1 (the symbolic link you can see above basically means that any program that uses the file /dev/modem is actually dealing with /dev/ttyS1).

Please remember that your system might be different, and you might not have a symbolic link that points to ttyS1 or ttyS0. In fact, the goal of this section is to have a symbolic link in /dev called modem that points to the right serial port (that is, the one your modem is connected to).

First we are going to determine to which serial port your modem is connected. Type the command **minicom**. To see if minicom is already talking to your modem, just type **at** and press Enter. If you see an "OK" response, minicom is using the right file in /dev to access the modem. Otherwise, for some reason minicom was unable to talk to the modem.

If there was no OK response from the modem, the first thing to do is to find out where the modem is (that is, to which serial port the modem is connected). This is done easily from minicom: press Ctrl-A and then O. Please note that sometimes minicom is configured so that it uses the Alt key. If that is the case, you should remember that every time I write Ctrl-*key*, you should press Alt-*key* (e.g., instead of Ctrl-A and then O, you should press Alt and then O).

The minicom configuration screen will pop up. Select "Serial port setup" and press Enter. Now, press A and choose a different serial port. For example if the current port is /dev/ttyS0, change it into /dev/ttyS1 (Figure 1). You will now have to exit minicom and run it again. Press Enter and select "Save default as dfl". Then select "Exit" (you will exit the configuration screen) and quit minicom (Ctrl-A and then Q).

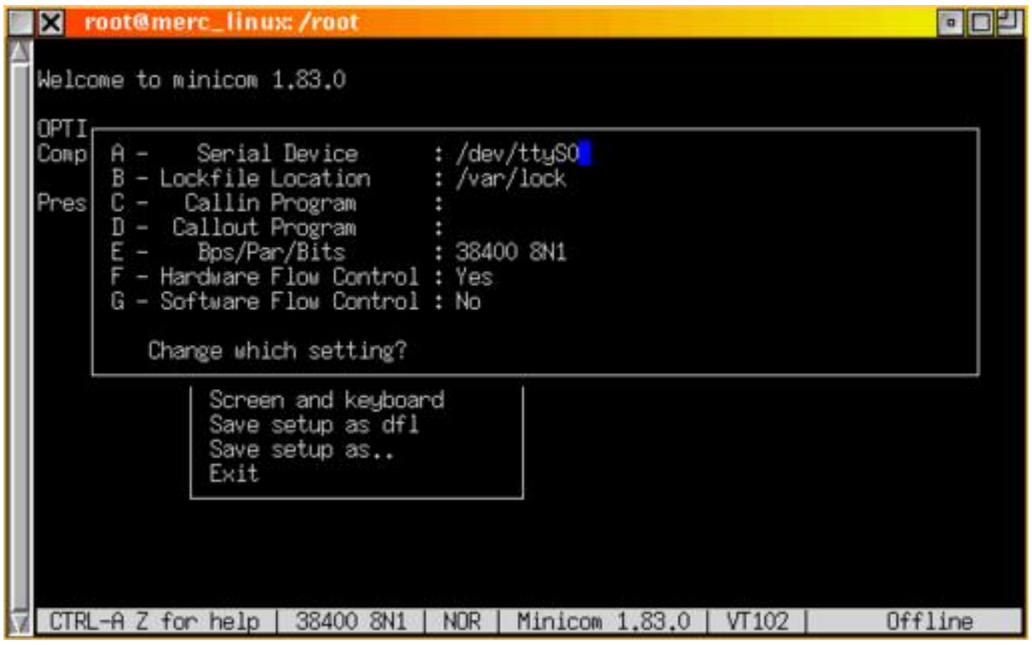
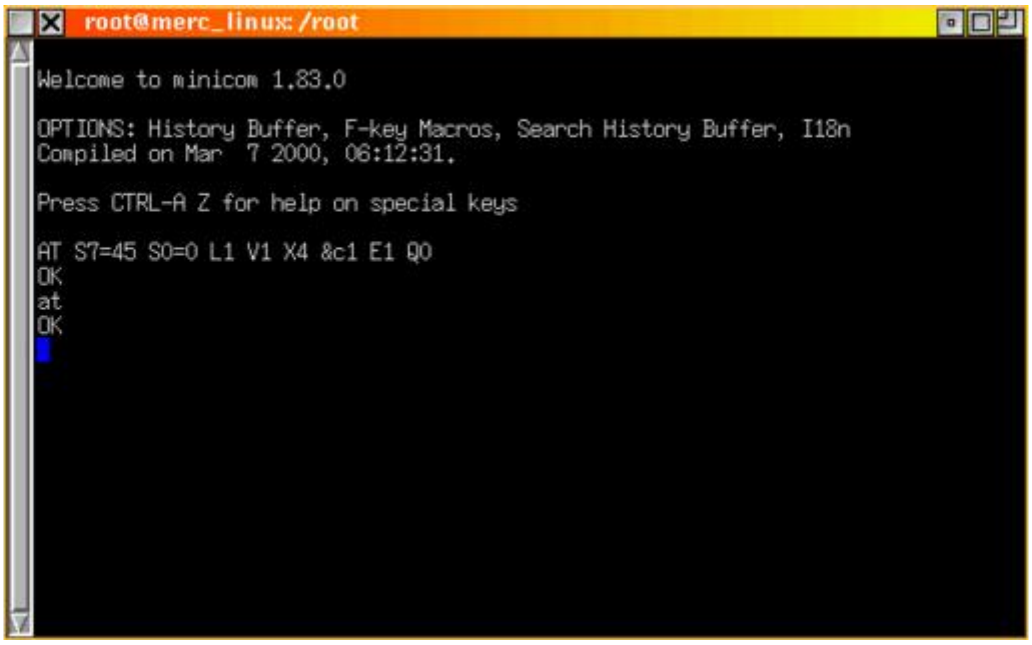

```
Welcome to minicom 1.83.0

OPTI
Comp  A -     Serial Device       : /dev/ttyS0
      B - Lockfile Location       : /var/lock
Pres  C -    Callin Program       :
      D -   Callout Program       :
      E -      Bps/Par/Bits       : 38400 8N1
      F - Hardware Flow Control   : Yes
      G - Software Flow Control   : No

         Change which setting?

            Screen and keyboard
            Save setup as dfl
            Save setup as..
            Exit

 CTRL-A Z for help | 38400 8N1 | NOR | Minicom 1.83.0 | VT102 |      Offline
```

Figure 1. From this screen you can tell minicom which serial port it should use to talk to the modem.

Now, run it again. **minicom** will talk to the serial port you chose. Type **at** and press Enter. If you receive an OK as a response, you have found the modem (see Figure 2). Otherwise, you will have to repeat the steps outlined in this section until you find the serial port (that is, when you receive an OK response from the modem when you type **at** after running minicom). Please remember that normal debugging procedures should be followed when you try to figure out where your modem is. In particular, make sure your modem is switched on, is connected to the computer, is using the right power supply and that its connection with the computer is firm.



```
Welcome to minicom 1.83.0

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Mar  7 2000, 06:12:31.

Press CTRL-A Z for help on special keys

AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
OK
at
OK
```

Figure 2. This is what minicom should look like when the modem responds with OK.

They seem to be trivial checks, but it is because of their triviality that they are often ignored. As a result, you can spend hours trying to configure a modem that actually is not connected or switched on.

If there is an OK response from the modem, everything is working, and it's only a matter of sorting a few things out. From minicom, press Ctrl-A and then O. The minicom configuration screen will pop up. Select “Serial port setup” and press Enter.

On the first line, you will be able to see what file minicom is using to communicate to the modem. If it is /dev/modem, everything is configured properly on your system, and you can skip to the next section of this article (“Talking to the Modem”).

If you received the OK response, but minicom is configured to use a different port (such as /dev/ttyS0), then you have to create a symbolic link called modem in /dev in order to make sure you have a working /dev/modem entry in your system (remember, this was the primary goal of this section). Write down which port minicom actually is using (let's suppose it is /dev/ttyS0). Quit the configuration screen (pressing Enter and selecting “Exit”), and quit minicom (by pressing Ctrl-A and then Q). Now you are back to your shell. Go into the /dev directory and create the symbolic link for “modem” like this:

```
cd /dev
ln -sf ttyS0 modem
ls -l modem
lrwxrwxrwx  1 root  root  5 Aug  3 12:32 modem -> ttyS0
```

Of course, you will have to substitute ttyS0 with the port you found in the minicom configuration screen.

Now, run minicom again. Make sure everything still works by typing **at** and pressing Enter. You should receive the reassuring OK response (see Figure 2). Go to the minicom configuration screen again (Ctrl-A and O) and select “Serial port setup”. Now change the default device by pressing A and substituting /dev/ttyS0 with /dev/modem (see Figure 1). Press Enter, then “Save setup as dfl” and select “Exit” to exit the configuration screen. In order to have minicom work with the new configuration you have to exit minicom (Ctrl-A and Q) and run it again.

Now minicom is using the device /dev/modem. Type **at**. You should receive an OK response (see Figure 2). If that is the case, congratulations!

Configuring the modem is the most critical step when you try to connect to the Internet. Many users do not know to which serial port their modem is connected. As shown in the previous section, it is clear that the modem was already configured; you only had to find out where it was and create a symbolic link in /dev that pointed to the right device file.

Talking to the Modem

It is interesting to notice that minicom itself doesn't have any idea about the at command or the OK response. In fact, minicom's main tasks are very simple: to display the characters that come from the serial port and to send the characters the user keys in through the keyboard to the serial port.

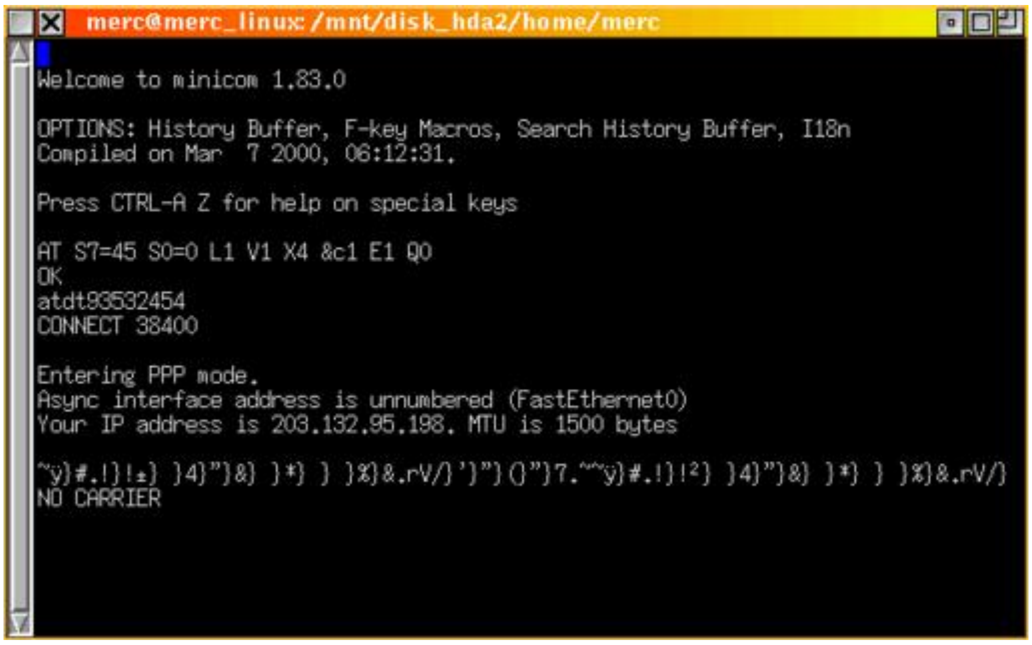
In a sense, the modem is like a robot that talks to the computer using the serial port. That is why using minicom allows you to, in a sense, have your own private conversation with the modem. In the previous section, for example, you sent the modem the string **at**<Enter>, and the modem responded with the string OK<Enter>.

What kind of conversation can you have with the modem? As a matter of fact, every modem understands the Hayes command set that is based on the at command set. Some manufacturers also offer some very interesting extensions. If you want to find out what they are, all you have to do is look up the commands available for your modem in your modem manual. For example, the commands ATL-1, ATL-2 and ATL-3 are common to every modem; they set the speaker volume to low, medium or high.

Even though you don't need to know many of these commands to use the modem effectively, you might want to try out a few commands to understand a bit more about how your modem works. Remember that it is always possible to type the commands **at&f** and **at&w** to reset the modem to the manufacturer's configuration in case you send the modem a few unsafe commands that compromise its configuration and prevent it from connecting.

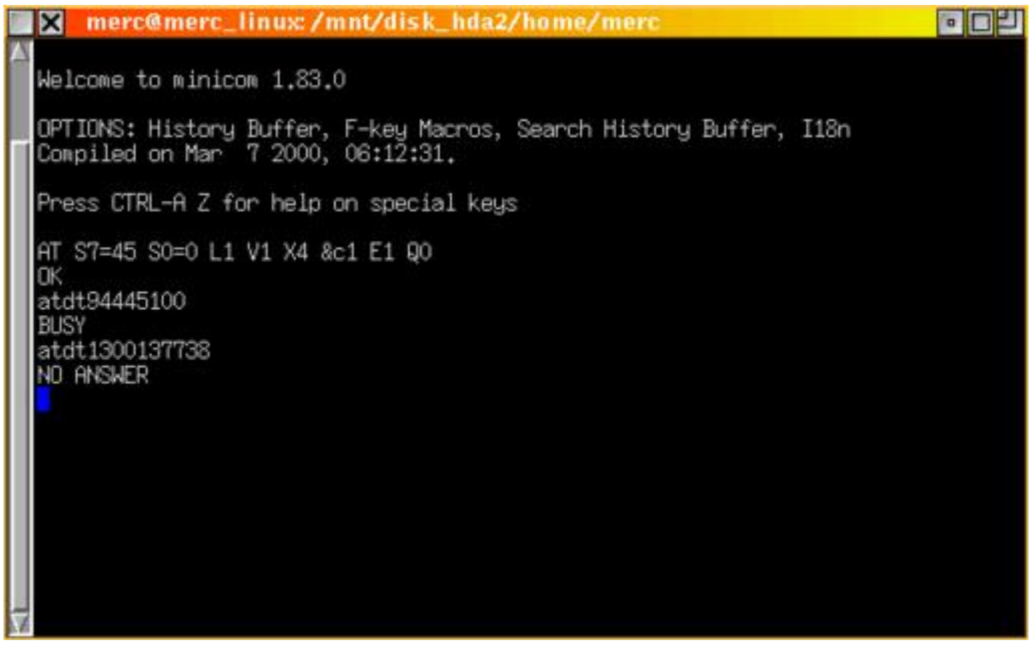
The command **ATDT93355100** is used to dial a number, in this case 93355100. DT stands for dial tone, as opposed to DP, which stands for dial pulse. You might have to use the latter if you live in a rural area and are connected to an analog telephone exchange. Try to send the modem the command **ATDT**, followed by the number of your internet service provider, to see what happens.

In my case, the connection looks like the one in Figure 3. As you can see, after the hardware handshake (with a lot of whistling on both sides), the modem responds with a nice CONNECT 52500 message, which means the connection was established without any trouble. Of course, there are other messages you may get as a response: NO DIALTONE (there is no phone line attached to the modem), BUSY (the line is engaged), NO ANSWER (there was no answer) and so on (see Figure 4).



```
Welcome to minicom 1.83.0

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Mar  7 2000, 06:12:31.

Press CTRL-A Z for help on special keys

AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
OK
atdt93532454
CONNECT 38400

Entering PPP mode.
Async interface address is unnumbered (FastEthernet0)
Your IP address is 203.132.95.198. MTU is 1500 bytes

~ÿ}#.!}!±} }4}"}&} }*} } }%}&.rV/}'}"}(}"}7.~~ÿ}#.!}!²} }4}"}&} }*} } }%}&.rV/}
NO CARRIER
```

Figure 3. My screen after I have established a successful connection to my ISP.



```
Welcome to minicom 1.83.0

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Mar  7 2000, 06:12:31.

Press CTRL-A Z for help on special keys

AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
OK
atdt94445100
BUSY
atdt1300137738
NO ANSWER
```

Figure 4. My screen after dialing two numbers. The first one was engaged, and the other one didn't answer.

In my case, the connection was established successfully and my ISP sent me the cryptic string as soon as I connected (the one that starts with ~}#.!}!±} }4}` `}&} }*} } }%). In fact, my ISP is expecting to have a conversation with my PPP dæmon, whereas I am only running minicom, which will show exactly what my ISP would have sent to my PPP dæmon.

It is very important to remember that from now on the modem is going to stop responding to any at commands; all the information sent to the serial port will be modulated and sent to the other side of the line. At the same time, all the information coming from the other side of the line will be demodulated and sent to the serial port (hence, the word modem). The computer (or, better, the serial port) is not aware of all this. It runs exactly as if there was a serial cable running from your computer to the computer on the other side. In fact, if you have two PCs at home, you can network them together quite easily using pppd and connecting their serial ports with a serial cable (you have to make sure it's an inverted cable so that the send pin of the first computer is connected to the receive pin of the second computer, and vice versa).

Conclusion

Next month I'll talk about configuring your computer so that you can connect to the Internet without using any advanced tools. Stay tuned!

Tony Mobily (merc@mobily.com) is the technical editor of Login, an Italian computer magazine. He is a Linux Certification Instructor (www.linuxcertification.com), a system administrator and is training as a dancer. He knows how to use English, Italian, C, Perl and a few other languages.

Archive Index Issue Table of Contents

Advanced search

Advanced search

Inside the Linux Packet Filter

Gianluca Insolvibile

Issue #94, February 2002

In Part I of this two-part series on the Linux Packet Filter, Gianluca describes a packet's journey through the kernel.

Network geeks among you may remember my article, “Linux Socket Filter: Sniffing Bytes over the Network”, in the June 2001 issue of *LJ*, regarding the use of the packet filter built inside the Linux kernel. In that article I provided an overview of the functionality of the packet filter itself; this time, I delve into the depths of the kernel mechanisms that allow the filter to work and share some insights on Linux packet processing internals.

Last Article's Points

In the previous article, some arguments regarding kernel packet processing were raised. It is worthwhile to recall briefly the most important of them:

- Packet reception is first dealt with at the network card's driver level, more precisely in the interrupt service routine. The service routine looks up the protocol type inside the received frame and queues it appropriately for later processing.
- During reception and protocol processing, packets might be discarded if the machine is congested. Furthermore, as they travel upward toward user land, packets lose network lower-level information.
- At the socket level, just before reaching user land, the kernel checks whether an open socket for the given packet exists. If it does not, the packet is discarded.
- Then the Linux kernel implements a generic-purpose protocol, called PF_PACKET, which allows you to create a socket that receives packets directly from the network card driver. Hence, any other protocols' handling is skipped, and any packets can be received.

- An Ethernet card usually passes only the packets destined to itself to the kernel, discarding all the others. Nevertheless, it is possible to configure the card in such a way that all the packets flowing through the network are captured, independent of their MAC address (promiscuous mode).
- Finally, you can attach a filter to a socket, so that only packets matching your filter's rules are accepted and passed to the socket. Combined with PF_PACKET sockets, this mechanism allows you to sniff selected packets efficiently from your LAN.

Even though we built our sniffer using PF_PACKET sockets, the Linux socket filter (LSF) is not limited to those. In fact, the filter also can be used on plain TCP and UDP sockets to filter out unwanted packets—of course, this use of the filter is much less common.

In the following, I sometimes refer either to a socket or to a sock structure. As far as this article is concerned, both forms indicate the same object, and the latter corresponds to the kernel's internal representation of the former. Actually, the kernel holds both a socket structure and a sock structure, but the difference between the two is not relevant here.

Another data structure that will recur quite often is the sk_buff (short for socket buffer), which represents a packet inside the kernel. The structure is arranged in such a way that addition and removal of header and trailer information to the packet data can be done in a relatively inexpensive way: no data actually needs to be copied since everything is done by just shifting pointers.

Before going on, it may be useful to clear up possible ambiguities. Despite having a similar name, the Linux socket filter has a completely different purpose with respect to the Netfilter framework introduced into the kernel in early 2.3 versions. Even if Netfilter allows you to bring packets up to user space and feed them to your programs, the focus there is to handle network address translation (NAT), packet mangling, connection tracking, packet filtering for security purposes and so on. If you just need to sniff packets and filter them according to certain rules, the most straightforward tool is LSF.

Now we are going to follow the trip of a packet from its very ingress into the computer to its delivery to user land at the socket level. We first consider the general case of a plain (i.e., not PF_PACKET) socket. Our analysis at link layer level is based on Ethernet, since this is the most widespread and representative LAN technology. Cases of other link layer technologies do not present significant differences.

Ethernet Card and Lower-Kernel Reception

As we mentioned in the previous article, the Ethernet card is hard-wired with a particular link layer (or MAC) address and is always listening for packets on its interface. When it sees a packet whose MAC address matches either its own address or the link layer broadcast address (i.e., FF:FF:FF:FF:FF:FF for Ethernet) it starts reading it into memory.

Upon completion of packet reception, the network card generates an interrupt request. The interrupt service routine that handles the request is the card driver itself, which runs with interrupts disabled and typically performs the following operations:

- Allocates a new sk_buff structure, defined in include/linux/skbuff.h, which represents the kernel's view of a packet.
- Fetches packet data from the card buffer into the freshly allocated sk_buff, possibly using DMA.
- Invokes netif_rx(), the generic network reception handler.
- When netif_rx() returns, re-enables interrupts and terminates the service routine.

The netif_rx() function prepares the kernel for the next reception step; it puts the sk_buff into the incoming packets queue for the current CPU and marks the NET_RX softirq (softirq is explained below) for execution via the __cpu_raise_softirq() call. Two points are worth noticing at this stage. First, if the queue is full the packet is discarded and lost forever. Second, we have one queue for each CPU; together with the new deferred kernel processing model (softirqs instead of bottom halves), this allows for concurrent packet reception in SMP machines.

If you want to see a real-world Ethernet driver in action, you can refer to the simple NE 2000 card PCI driver, located in drivers/net/8390.c; the interrupt service routine called ei_interrupt(), calls ei_receive(), which in turn, performs the following procedure:

- Allocates a new sk_buff structure via the dev_alloc_skb() call.
- Reads the packet from the card buffer (ei_block_input() call) and sets skb->protocol accordingly.
- Calls netif_rx().
- Repeats the procedure for a maximum of ten consecutive packets.

A slightly more complex example is provided by the 3COM driver, located in 3c59x.c, which uses DMA to transfer the packet from the card memory to the sk_buff.

Network Core Processing

Let's take a closer look at the netif_rx() function. As mentioned before, this function has the task of receiving a packet from a network driver and queuing it for upper-layer processing. It acts as a single gathering point for all the packets collected by the different network card drivers, providing input to the upper protocols' processing.

Since this function runs in interrupt context (that is, its execution flow follows the interrupt service path) with other interrupts disabled, it has to be quick and short. It cannot perform lengthy checks or other complex tasks since the system is potentially losing packets while netif_rx() runs. So, what this function does is basically select the packet queue from an array called softnet_data, whose index is based on the CPU currently running. It then checks the status of the queue, identifying one of five possible congestion levels: NET_RX_SUCCESS (no congestion), NET_RX_CN_LOW, NET_RX_CN_MOD, NET_RX_CN_HIGH (low, moderate and high congestion, respectively) or NET_RX_DROP (packet dropped due to critical congestion).

Should the critical congestion level be reached, netif_rx() engages a throttling policy that allows the queue to go back to a noncongested status, avoiding service disruption due to kernel overload. Among other benefits, this helps avert possible DOS attacks.

Under normal conditions, the packet is finally queued (__skb_queue_tail()), and __cpu_raise_softirq(cpuid, NET_IF_SOFTIRQ) is called. The latter function has the effect of scheduling a softirq for execution.

The netif_rx() function terminates, returning a value indicating the current congestion level to the caller. At this point, interrupt context processing is done, and the packet is ready to be taken care of by upper-layer protocols. This processing is deferred to a later time, when interrupts will have been re-enabled and execution timing will not be as critical. The deferred execution mechanism has changed radically from kernel versions 2.2 (where it was based on bottom halves) to versions 2.4 (where it is based on softirqs).

softirqs vs. Bottom Halves

Explaining in detail about bottom halves (BHs) and their evolution is out of the scope of this article. But, some points are worth recalling briefly.

First off, their design was based on the principle that the kernel should perform as few computations as possible while in interrupt context. Thus, when long operations were to be done in response to an interrupt, the corresponding driver would mark the appropriate BH for execution, without actually doing anything complex. Then, at a later time, the kernel would have checked the BH mask to determine whether some BHs were marked for execution and execute them before any application-level task.

BHs worked quite well, with one important drawback: due to their structure, their execution was serialized strictly among CPUs. That is, the same BH could not be executed by more than one CPU at the same time. This obviously prevented any kind of kernel parallelism on SMP machines and seriously affected performance. **softirqs** represent the 2.4-age evolution of BHs and, together with tasklets, belong to the family of kernel software interrupts, pieces of code that can be executed by the kernel when requested, without strict response-time guarantees.

The major difference with respect to BHs is that the same softirq may be run on more than one CPU at a time. Serialization, if required, now must be obtained explicitly by using kernel spinlocks.

softirq's Internals

softirq's processing core is performed in the do_softirq() routine, located in kernel/softirq.c. This function checks a bit mask, and if the bit corresponding to a given softirq is set, it calls the appropriate handling routine. In the case of NET_RX_SOFTIRQ, the one we are interested in at this time, the relevant function is net_rx_action(), located in net/core/dev.c. The do_softirq() function may get called from three distinct places inside the kernel: do_IRQ(), in kernel/irq.c, which is the generic interrupt handler; system calls' exit point, in kernel/entry.S; and schedule(), in kernel/sched.c, which is the main process scheduling function.

In other words, execution of a softirq may happen either when a hardware interrupt has been processed, when an application-level process invokes a system call or when a new process is scheduled for execution. This way, softirqs are drained frequently enough that none of them will lie waiting for their turn for too long.

The trigger mechanism also was exactly the same for the old-style bottom halves.

The NET_RX softirq

We've seen the packet come in through the network interface and get queued for later processing. Then, we've considered how this processing is resumed by a call to the net_rx_action() function. It's now time to see what this function does. Basically, its operation is pretty simple: it just dequeues the first packet (sk_buff) from the current CPU's queue and runs through the two lists of packet handlers, calling the relevant processing functions.

Some more words are worth spending on those lists and how they are built. The two lists are called ptype_all and ptype_base and contain, respectively, protocol handlers for generic packets and for specific packet types. Protocol handlers register themselves, either at kernel startup time or when a particular socket type is created, declaring which protocol type they can handle; the involved function is dev_add_pack() in net/core/dev.c, which adds a packet type structure (see include/linux/netdevice.h) containing a pointer to the function that will be called when a packet of that type is received. Upon registration, each handler's structure is either put in the ptype_all list (for the ETH_P_ALL type) or hashed into the ptype_base list (for other ETH_P_* types).

So, what the NET_RX softirq does is call in sequence each protocol handler function registered to handle the packet's protocol type. Generic handlers (that is, ptype_all protocols) are called first, regardless of the packet's protocol; specific handlers follow. As we will see, the PF_PACKET protocol is registered in one of the two lists, depending on the socket type chosen by the application. On the other hand, the normal IP handler is registered in the second list, hashed with the key ETH_P_IP.

If the queue contains more than one packet, net_rx_action() loops on the packets until either a maximum number of packets has been processed (netdev_max_backlog) or too much time has been spent here (the time limit is 1 jiffy, i.e., 10ms on most kernels). If net_rx_action() breaks the loop leaving a non-empty queue, the NET_RX_SOFTIRQ is enabled again to allow for the processing to be resumed at a later time.

The IP Packet Handler

The IP protocol receive function, namely ip_rcv() (in net/ipv4/ip_input.c), is pointed to by the packet type structure registered within the kernel at startup time (ip_init(), in net/ipv4/ip_output.c). Obviously, the registered protocol type for IP is ETH_P_IP.

Thus, ip_rcv() gets called from within net_rx_action() during the processing of a softirq, whenever a packet with type ETH_P_IP is dequeued. This function performs all the initial checks on the IP packet, which mainly involve verifying its integrity (IP checksum, IP header fields and minimum significant packet length). If the packet looks correct, ip_rcv_finish() is called. As a side note, the call to this function passes through the Netfilter prerouting control point, which is practically implemented with the NF_HOOK macro.

ip_rcv_finish(), still in ip_input.c, mainly deals with the routing functionality of IP. It checks whether the packet should be forwarded to another machine or if it is destined to the local host. In the former case, routing is performed, and the packet is sent out via the appropriate interface; otherwise, local delivery is performed. All the magic is realized by the ip_route_input() function, called at the very beginning of ip_rcv_finish(), which determines the next processing step by setting the appropriate function pointer in skb->dst->input. In the case of locally bound packets, this pointer is the address of the ip_local_deliver() function. **ip_rcv_finish()** terminates with a call to skb->dst->input().

At this point, packets definitely are traveling toward the upper-layer protocols. Control is passed to ip_local_deliver(); this function just deals with IP fragments' reassembly (in case the IP datagram is fragmented) and then goes over to the ip_local_deliver_finish() function. Just before calling it, another Netfilter hook (the ip-local-ip) is executed.

The latter is the last call involving IP-level processing; ip_local_deliver_finish() carries out the tasks still pending to complete the upper part of layer 3. IP header data is trimmed so that the packet is ready to be transferred to the layer 4 protocol. A check is done to assess whether the packet belongs to a raw IP socket, in which case the corresponding handler (raw_v4_input()) is called.

Raw IP is a protocol that allows applications to forge and receive their own IP packets directly, without incurring actual layer 4 processing. Its main use is for network tools that need to send particular packets to perform their tasks. Well-known examples of such tools are ping and traceroute, which use raw IP to build packets with specific header values. Another possible application of raw IP is, for example, realizing custom network protocols at the user level (such as RSVP, the resource reservation protocol). Raw IP may be considered a standard equivalent of the PF_PACKET protocol family, just shifted up one open systems interconnection (OSI) level.

Most commonly, though, packets will be headed toward a further kernel protocol handler. In order to determine which one it is, the Protocol field inside the IP header is examined. The method used by the kernel at this point is very similar to the one adopted by the net_rx_action() function; a hash is defined, called inet_protos, which contains all the registered post-IP protocol handlers. The hash key is, of course, derived from the IP header's protocol field. The inet_protos hash is filled in at kernel startup time by inet_init() (in net/ipv4/af_inet.c), which repeatedly calls inet_add_protocol() to register TCP, UDP, ICMP and IGMP handlers (the latter only if multicast is enabled). The complete protocol table is defined in net/ipv4/protocol.c.

For each protocol, a handler function is defined: tcp_v4_rcv(), udp_rcv(), icmp_rcv() and igmp_rcv() are the obvious names corresponding to the above-mentioned protocols. One of these functions is thus called to proceed with packet processing. The function's return value is used to determine whether an ICMP Destination Unreachable message has to be returned to the sender. This is the case when the upper-level protocols do not recognize the packet as belonging to an existing socket. As you will recall from the previous article, one of the issues in sniffing network data was to have a socket able to receive packets independent of their port/address values. Here (and in the just-mentioned *_rcv() functions) is the point where that limitation arises from.

Conclusion

At this point, the packet is more than halfway through its journey. Since space is limited in our beloved magazine, we will leave the packet in the capable hands of upper-layer 3 protocols until next month. What still remains to be explored is layer 4 processing (TCP and UDP), PF_PACKETs handling and, of course, the socket filter hooks and implementation. Be patient!

Resources

Creation of PF_PACKET Sockets

Gianluca Insolvibile has been a Linux enthusiast since kernel 0.99pl4. He currently deals with networking and digital video research and development. He can be reached at g.insolvibile@cpr.it.

Introducing Zope

Reuven M. Lerner

Issue #94, February 2002

Reuven gives a whirlwind tour of the open-source Zope application server, beginning with an exploration of Zope as a web development platform.

No matter what language and operating system you use, web development typically means working with HTML files, graphic files, standalone programs, hybrid HTML/code templates and a connection to a relational database. Experienced developers can move easily from Perl to PHP to ASP because the concepts translate from one language to another with only minor variations. This convergence of paradigms has been convenient for developers, at the cost of complacency and laziness in the web development community.

Luckily, the open-source Zope application server is there to shake us from our complacency and open our eyes to new ways in which we can develop web applications. Zope, written and supported by Zope Corporation (formerly Digital Creations), changes everything that you ever knew about web development. You still can create dynamically generated content and work with relational databases, but Zope does so in very different ways from every other application server on the market.

This month, we begin to explore Zope as a web development platform. Along the way, I hope it becomes obvious that while Zope is different from other environments, its elegance and power make it a strong contender for open-source enthusiasts.

What Is Zope?

Part of the difficulty in understanding Zope is that it's actually several things at once. Simply put, Zope provides you with everything you need to create web applications. Installing Zope gives you a simple HTTP, FTP and Web-DAV server (known as the ZServer), an object database (known as ZODB) and a framework for creating server-side web applications.

Most of Zope is written in Python, which means that it is ported across platforms easily. And while it makes sense that a program written in Perl, Python or Java can run equally on Linux and Windows easily, it is still unusual for a prominent open-source system to run on Windows.

A typical web site has a mix of static HTML files, templates and graphic files. When the HTTP server receives a request, the server determines what kind of file was requested (by looking at the file's extension, location and MIME type), executes any necessary code inside the file and returns an HTTP response.

In Zope, nothing is stored on the filesystem. Rather, all site content and programs are stored inside of the ZODB object database. (The database itself typically is stored as a large disk file, which can be backed up or copied to backup Zope servers.)

To create a simple HTML file, you will need to create a "file" object in ZODB. To display an image on the web site, you will need to create an image object in ZODB. And to execute a piece of code when a certain URL is invoked, you must store the code in ZODB.

Luckily, storing and retrieving items in ZODB is quite easy. Zope takes care of most of this for you, allowing you to set high-level parameters at installation time. Moreover, Zope is designed so that it can be controlled completely via one web browser. You can use one of Zope's web-based tools, which makes it possible to create, edit and administer a web site using nothing more than your web browser. So you can create, modify or delete any content on the site (HTML, images or directories) using your browser.

If you prefer to edit text (and code) in Emacs rather than in a web text area, the ZServer supports FTP, displaying the objects in ZODB as if they were in a filesystem hierarchy. If you (and the nontechnical staff that you trained) previously were using FTP to transfer files to and from your web server, ZServer's FTP implementation will allow you to continue along that path almost seamlessly.

Installing Zope

For all of its complexity, Zope is surprisingly easy to install. You will need to download the latest version from www.zope.org (see Resources). After you choose the version that you want to download, switch into your Zope directory (typically /usr/local/zope/, but anything will do), and unpack the Zope file:

```
mkdir /usr/local/zope
cd /usr/local/zope
tar zxvvf /downloads/Zope-2.4.3-linux2-x86.tgz
```

Opening up the Zope archive reveals a number of directories, including:

- bin, where the Zope startup and shutdown scripts are located.
- doc, where the complete Zope documentation is kept.
- lib, where Python and all of the various Zope applications (known as products) are kept.
- ZServer, which contains the classes necessary for the ZServer.
- utilities, which contains several Zope-related utilities.

Before you can start Zope for the first time, you must install it with the install script. Only run this script after you have placed the Zope files in their final location, since install uses the current directory name to set several global configuration values.

After you have finished installing Zope, you can start it up by running the start script in the main Zope directory. By default, this script starts Zope's HTTP server on port 8080 (with an FTP server running on another port). You can change these values with a command-line switch; a full list is available by typing **start -help**.

While Zope is written largely in Python, Python does not need to be installed on your system. Zope comes with its own copy of Python, which it uses instead of whatever is on the operating system. This decreases the chances for version problems and errors. As of this writing, Zope uses Python 2.1 (the latest stable version), but Python 2.2 should be coming out in the near future. It will be interesting to see when Zope adopts it.

The install script not only sets up the initial Zope configuration, but also creates a hard-to-guess password for the system's “admin” user. We will need this password in just a few minutes, so be sure to write it down.

Administering Zope

Now that we have started Zope, how do we use it? Well, the first and most important step is to fire up our web browser and point it to our computer, giving it the URL http://localhost:8080/. This displays some initial Zope information, including pointers to documentation and web sites.

Let's leave this introductory screen behind, moving instead to the main Zope management interface at http://localhost:8080/manage. You will be prompted to enter the username and password for a site manager; use the admin user and the password that the install script printed out when you first installed Zope.

Once you have entered the correct password, your browser window will be divided into two vertical frames: the left-hand side displays the hierarchy of your Zope server, while the right-hand side displays the details of the object you are currently viewing.

Creating Basic Content

It's easy to understand what happens when you ask an Apache server for the document /foo/bar. Apache looks for a file named bar in the foo directory underneath its document root. If such a file exists, then Apache reads it from disk and returns its content in an HTTP response. If the file is a CGI program, then it executes the program and returns its output in an HTTP response. And if the file does not exist, Apache returns an error message to the user's browser.

Zope interprets URLs quite differently; the URL /foo/bar is interpreted as a request for Zope to invoke a display method on the bar object inside of the foo object. If either foo or bar does not exist, then Zope will return an error message indicating that no such object can be found in ZODB.

If we always had to worry about creating objects and writing display methods, then Zope would be a high-powered toy for programmers. Luckily for us, Zope lets us almost pretend that ZODB is a hierarchical filesystem into which we are placing our HTML files and graphics.

For example, we can create a simple HTML file using nothing more than our web browser. While inside of Zope's root folder (to which you can always return by clicking in the upper left-hand side of the browser's left frame), create a new document by choosing DTML Document from the “Select type to add...” selection list in the top right-hand corner. (As we will soon see, DTML—Document Template Markup Language—is Zope's superset of HTML; Zope almost always refers to DTML documents rather than HTML documents.)

Setting DTML Parameters

Your web browser should now display a short HTML form with three elements:

- The id text field. Every object in Zope has an ID, and IDs must be unique within a folder. IDs are analogous to filenames on a disk, and they are used to identify objects within a URL. In the URL /foo/bar, foo is the ID of a folder object, while bar is the ID of an object within that folder. Zope documents traditionally don't have suffixes (e.g., .html and .gif); because the system knows what type of object is associated with each ID, such a suffix would be redundant.

- The title text field. The object's ID appears in URLs, but the object's title appears in all of the internal Zope management screens. (The object's title has nothing to do with the HTML <title> tag, which you still will have to create.)
- The file element allows you to upload a file from your local computer using HTTP uploads. This means that you can create a DTML file on your local computer and upload it to Zope when you are ready for testing.

For the purposes of this example, we will enter an ID of testdoc, and a title of “Test document”. While we simply could add this document to ZODB, that would leave it empty. Thus, we will click on the “Add and edit” button, which gives us a text area with some default content:

```
<dtml-var standard_html_header>
<h2><dtml-var title_or_id></h2>
<p>
This is the <dtml-var id> Document.
</p>
<dtml-var standard_html_footer>
```

You can edit the contents however you want within this text area. When you have finished, click on the save changes button at the bottom of the screen. This button brings you back to the editing window but adds a reminder indicating when the document was last modified.

Working with DTML

Our DTML document looks very much like HTML, except that it includes several tags that begin with <dtml-var>. DTML is actually a programming language that lets you do all sorts of things, but it is probably easiest (and best) to think of it as a very strong server-side include mechanism. As demonstrated in this sample document, <dtml-var> lets you insert dynamic values into the current document. Thus <dtml-var standard_html_header> inserts the contents of the object with the ID of standard_html_header, while <dtml-var id> inserts the ID of the current document. As you can imagine, <dtml-var title_or_id> displays either the title or the ID, depending on whether a title has been defined.

When Zope encounters the expression <dtml-var standard_html_header>, it looks in the current folder for an object with the ID standard_html_header. If such an object exists, then the <dtml-var> tag is replaced with the object's viewable contents. If no such object exists, then Zope repeats its search in the enclosing folder. In this way, Zope works its way up the entire hierarchy until it reaches the root folder.

This means that automatically inserted headers and footers depend not only on our <dtml-var> tags, but also on the location of our DTML document. The notion that an object's location in the object hierarchy affects its output is known as acquisition in the Zope world, and it is both important and useful. Using acquisition, you can have a different set of headers and footers for each folder. Moving a document from one folder to another changes the search path that Zope uses when looking for headers, footers and other objects. The behavior of an object in Zope thus depends not only on its definition, but also on its location in the object hierarchy. For this reason, acquisition is sometimes explained as analogous to the “nature vs. nurture” debate in biology: an object's definition can be seen as “nature”, while its location in the object hierarchy can be seen as “nurture”.

To view the current contents of the document, click on the view tab at the top of the frame. You will see the document's contents exactly as a user would want. To edit the document once again, I find it easiest to click on the “Root folder” link in the top-left corner, select the testdoc object and then edit things once more using the text area.

If you make a mistake while editing a DTML document, you will be happy to know that Zope provides infinite undo. Click on the undo tab on the top right of the DTML editing screen, and select the version to which you want to return. Infinite undo is one of my favorite features in Zope, not only because it allows me to undo my mistakes, but because it's so easily accessible to nonprogrammers, and it works with any kind of object on the system.

Other DTML Methods

There are a number of DTML tags, each of which begins with dtml-. Most DTML tags take one or more parameters, where each parameter allows it to work in a slightly different way. For example, we can modify our test document to include a simple use of <dtml-var expr>:

```
<p><dtml-var expr="5+10"></p>
```

The value of our expr in the above code is a simple Python expression. Zope evaluates the expression and replaces the dtml-var tag with the result of that evaluation. We also can try something more interesting than simple addition. For example, we can use the capitalize method in the string module (automatically imported into Zope) by naming it in the expr attribute:

```
<p><dtml-var expr="_.string.capitalize('abc')"></p>
```

Notice how we cannot invoke string.capitalize(), but must rather invoke _.string.capitalize().

While DTML allows you to use the string module in this way, you may never have to invoke it directly because so many useful string functions are built into DTML.

DTML was made for designers and other nonprogrammers to be able to create their own dynamic content without being stuck with the poor syntax and documentation of server-side includes. DTML is not as easy for nonprogrammers to learn as some of us might think, but it is far easier than teaching them a full-fledged programming language. And the fact that Zope performs some error checking when saving the DTML document helps to avoid some of the issues associated with runtime languages.

Conclusion

This month, we took a whirlwind tour of several of Zope's features, including through-the-web editing, the management interface, acquisition and simple DTML documents. Next month, we will look at Zope products—how to download and install them, and then how to write our own.

Resources

Reuven M. Lerner owns a small consulting firm specializing in web and internet technologies. He lives with his wife Shira and daughter Atara Margalit in Modi'in, Israel. You can reach him at reuven@lerner.co.il or on the ATF home page, www.lerner.co.il/atf.

Observe, Mon Cher Ami…

Marcel Gagné

Issue #94, February 2002

This month, Marcel introduces Gqcam, xawtv and MASH, some simple and fun tools to use with webcams, and Francois falls in love.

Cooking with Linux

Observe, *Mon Cher Ami*

This month Marcel introduces Gqcam, xawtv and MASH, some simple and fun tools to use with webcams.

by Marcel Gagné

Mon Dieu! This is terrible. Why did you not tell me, François? Surely it must have been obvious to you when I was putting the menu together. I must come up with something and quick, before our guests arrive. Too late, François. They are here.

Bonjour, *mes amis*. Welcome to *Chez Marcel*, home of fine Linux cooking. François, please, show our friends to their table. Forgive me if I seem a bit distressed, but I have been caught off guard today. You see, when François told me that the theme of the issue would be SOHO, I thought, *mais*, *c'est fantastique*! An issue featuring the wonders of our solar system. Imagine my surprise when I learned that SOHO, in this case, was Small Office/Home Office rather than Solar and Heliospheric Observatory. Then, I started playing with the words, you see. Observatory made me think of the technology for observing the stars, like that wonderful telescope named after Monsieur Hubble. And then I thought, but the true stars are this restaurant's fine guests—you, *mes amis*.

This brings us to tonight's menu. But first, we must have *du vin*, *non*? François, please bring up the 1998 Picadilly Valley Chardonnay from the cellar for our guests.

In order to achieve the highest level of success with the following recipes, a recent kernel is recommended highly. This is particularly true because a number of inexpensive webcams that you will find at your local electronics warehouse are USB devices. The more recent the kernel, the more likely that device will be supported. While taste-testing these recipes, I used a 2.4.9 kernel and two different webcams, one based on the CPiA chipset, and the other using a ov511 module.

The prevalent use for the webcam is on a web site, providing glimpses into the life of the individual running the site. Some sites provide a camera to reassure us that they are indeed working. Others are there to let parents observe their children playing at day care. The software that captures these images is sometimes called a frame grabber.

Ah, François, you are back. Please pour for our guests. François? François, I am talking to you. What do you mean you did not hear me? Ah, I see. A beautiful and mysterious lady is sitting at table 22. Pour the wine and then you may introduce yourself. *Vite*!

Where was I? *Mais oui*, frame grabbers. Gqcam is just such a package. With it you can set up a camera, keep an eye on whatever needs watching and capture images at will. You even can set it up to grab an ongoing sequence of images automatically, at whatever interval you choose. To get started with Gqcam, visit Cory Lueninghoener's web site (cse.unl.edu/~cluening/gqcam) to pick up the latest source, then extract and build it:

```
tar -xzvf gqcam-0.9.tar.gz
cd gqcam-0.9
make
```

You were waiting for something else, *non*? There is no **make install** for this one, *mes amis*. You can copy the final binary to /usr/local/bin manually, or wherever you want to save the executable. You also can execute the program directly from the build directory, if you wish, by typing **./gqcam &**. You may find, as I did, that the program looks to /dev/video by default. With the USB device, my webcam was actually at /dev/video0 while /dev/video was a directory. To get around this, I used the -v flag:

```
gqcam -v /dev/video0
```

What you choose to observe with Gqcam is entirely up to you, but as to what you can do with the program itself, allow me a few suggestions. If you look at

Figure 1, you'll see a shot of Gqcam in action. While it is running, you can do some interesting things. For instance, you can, at any time, click on the Snap Picture button to capture a frame. You'll be given the option of saving it in either PNG, PPM or JPEG format. If you were to click a number of images quickly, you could assemble them into a movie of sorts.

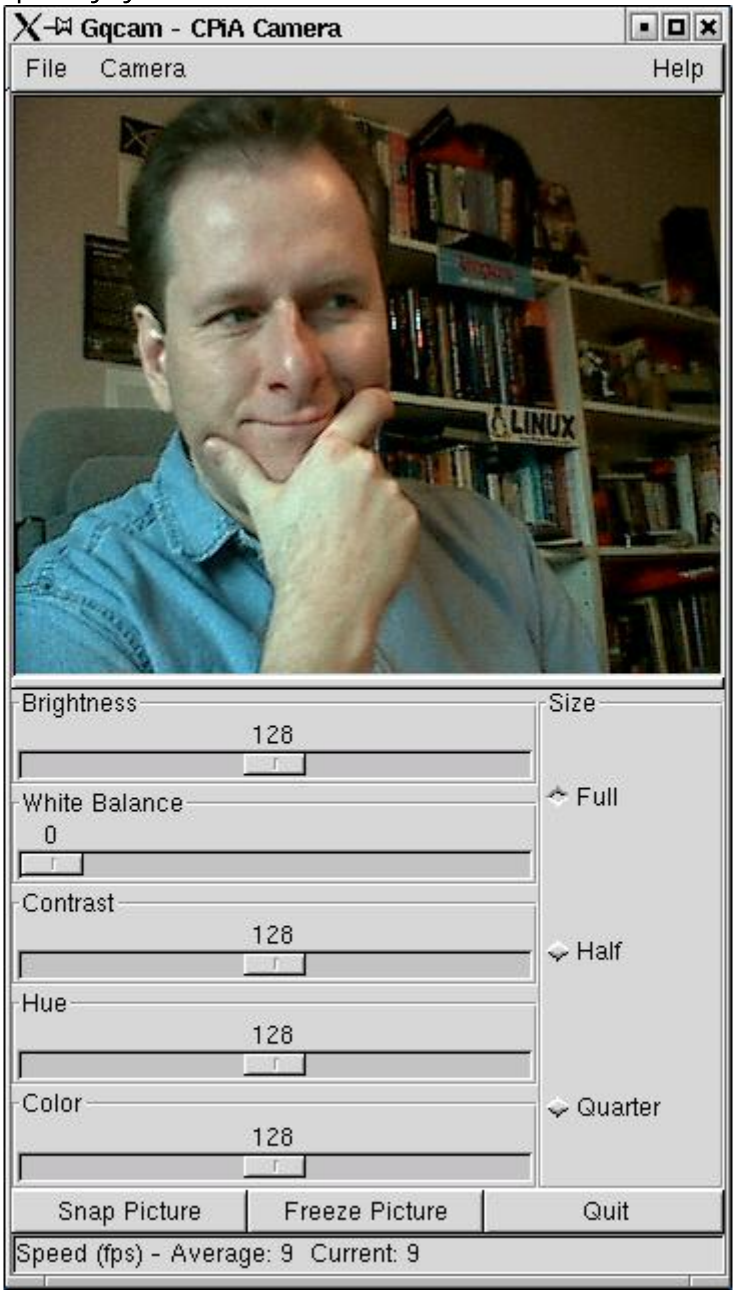


Figure 1. Gqcam Captures Our French Chef in a Casual Moment

Of course, it is tedious, is it not, to be snapping pictures of yourself continually. That's why Gqcam lets you set a timer. Click on Camera, then select Set timer. You can choose a directory where the image will be saved and an appropriate filename to save that image to, something like /somedir/mypic.jpg. Set the timer for one second and let it go. You'll end up with a number of images in /somedir that have the filename mypic followed by a period, a date and timestamp, and the .jpg extension. The fun part here is that you can run this little movie with a simple command that is part of the ImageMagick package (likely already installed on your machine):

```
animate mypic.*.jpg
```

Gqcam also can run directly from the command line, making it fairly simple to set up a cron job that will capture frames at some selected interval:

```
gqcam -v /dev/video0 -d /wwwdir/filename.png
```

The -d option specifies a location to dump a captured image. The default format for the image is PNG, but you can also use the -t flag to specify JPEG as an override. Now, if I happen to be running a web server somewhere, I could, when the mood takes me, snap a current picture and, *voilà*, we are all stars of the Internet, *non*? All you need now is a simple web page to display the image, something like this:

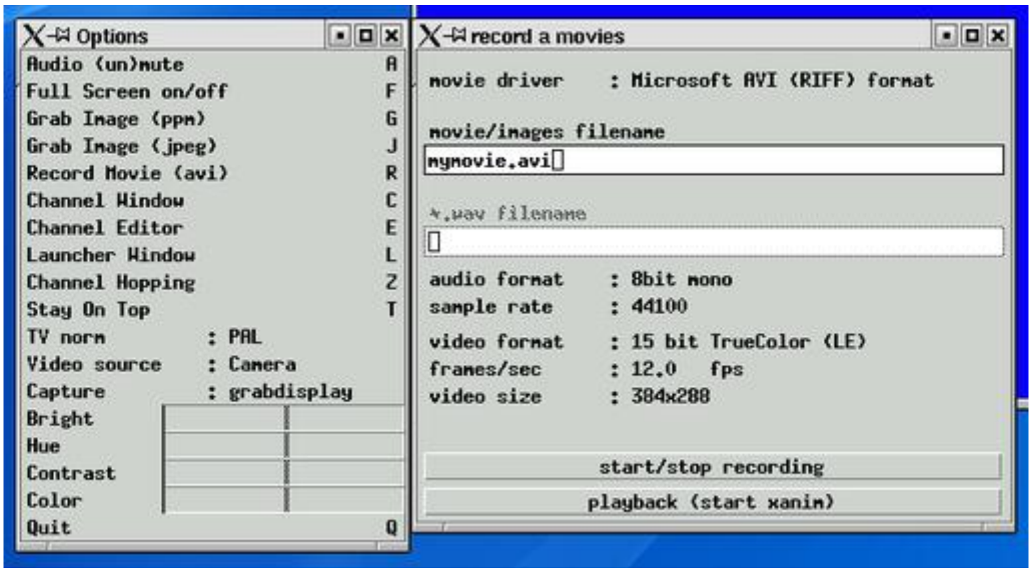
```
<html>
<head>
   <title>Cam Chez Marcel</title>
</head>
<body bgcolor="White">
<center>
   <h1>Bienvenue! You are watching Cam Chez Marcel</h1>
   <img src="camsnap.png">
</center>
</body>
</html>
```

Another program that performs some of the same functions as Gqcam is Gerd Knorr's xawtv. As its name implies, this is meant to provide you with a means of viewing television programs (with the appropriate hardware), but it also handles input from webcams. When you feel that you've done something that warrants observation on a regular basis (a video sales presentation, for instance), you may decide that providing a video clip adds a certain something extra. **xawtv** can give you that capability. Start by visiting the web site at bytesex.org/xawtv and picking up the latest copy. From a work directory, extract the source and build it:

```
tar -xzvf xawtv_3.64.tar.gz
cd xawtv-3.64
./configure
make
make install
```

You run the program by typing **xawtv &** at the command line. The program automatically sensed the location of my camera in this case, so no additional settings were necessary. When you right-click on the image, an options menu will appear that lets you change the video settings, such as brightness and color. Like Gqcam, you also can grab an image and save it in PPM or JPEG format. What's more interesting to this discussion is the next menu item.

If you press R, a recording window appears. The default is to save multiple images that you would then assemble using an external program. You can, however, click on movie driver and choose raw video data or an AVI format file. Choose AVI, select a filename to save it to, then click on the start recording tab. When you have the footage you want, click the tab again. (It actually says start/stop.) If you have the xanim package loaded, you may choose to play back what you recorded by clicking the tab immediately below. Have a look at Figure 2 for a sample of a recording session dialogue.



```
TV norm         : PAL
Video source    : Camera
Capture         : grabdisplay
```

```
movie driver     : Microsoft AVI (RIFF) format

movie/images filename
mymovie.avi

*.wav filename

audio format     : 8bit mono
sample rate      : 44100
video format     : 15 bit TrueColor (LE)
frames/sec       : 12.0   fps
video size       : 384x288
```

Figure 2. Setting up an AVI Capture Session with xawtv

Come closer, *mes amis*. Notice how François looks a little uncomfortable. He is somewhat taken with *la dame mystérieuse* at table 22. The poor boy is too shy to go talk to her in person. It is for my faithful waiter and others like him that I present you with our final recipe of the day.

When you want to be face to face, but distance (or other circumstances) makes it difficult, then video conferencing is the answer. This is where MASH comes into play. MASH is actually a collection of tools produced by the Open MASH Consortium (created by Larry Rowe and Steve McCanne). Their purpose was to create a public domain toolkit for developing collaboration and streaming applications. MASH stands for multimedia architecture that scales across heterogenous environments. No, really. It is all true.

If you feel so inclined, you can download the source and build the package from scratch:

```
tar -xzvf mash-src-5.1.5.tar.gz
cd mash-code
./build
```

The easiest method is to download the precompiled binary package from the Open MASH web site at www.openmash.org. Once that is done, you merely extract the package and run the installation script. Observe, *mes amis*:

```
   tar -xzvf mash-bin-5.1.5-linux-gnu.tar.gz
   cd mash-5.1.5
   ./setup.sh
```

So simple, *non*? As I mentioned, this is a collection of tools rather than a single program. The most interesting one however, particularly for François, is vic, the video-conferencing tool that allows for multiple users to participate in video communication:

```
   vic hostname/port_no
```

For instance, if I wanted to connect to a machine called speedy on my network on port 2002, I would issue the command like this:

```
   vic speedy/2002
```

You should see a small window pop up at this point with a simple message, “Waiting for video...”, in the center of it. The video you are waiting for, of course, is the young lady at table 22. For the young ladies out there, it may well be François. Who knows? While we wait for the other side of the connection, click the Menu button. A control window will appear that gives you control over many aspects of the current session. Look at the top of that window and click on the Transmit button. Even without the other side of the connection, you will see a small image of yourself sitting behind your camera.

Now, click on the small image. A somewhat larger session image appears. When the person on the other end comes on-line, their image will appear, and you can follow the same procedure to detach a more appetizing view. Figure 3 shows François talking to the mysterious lady using vic. As you can see from the image, he even is afraid to let his own image be seen and has provided an avatar of sorts. Poor François.

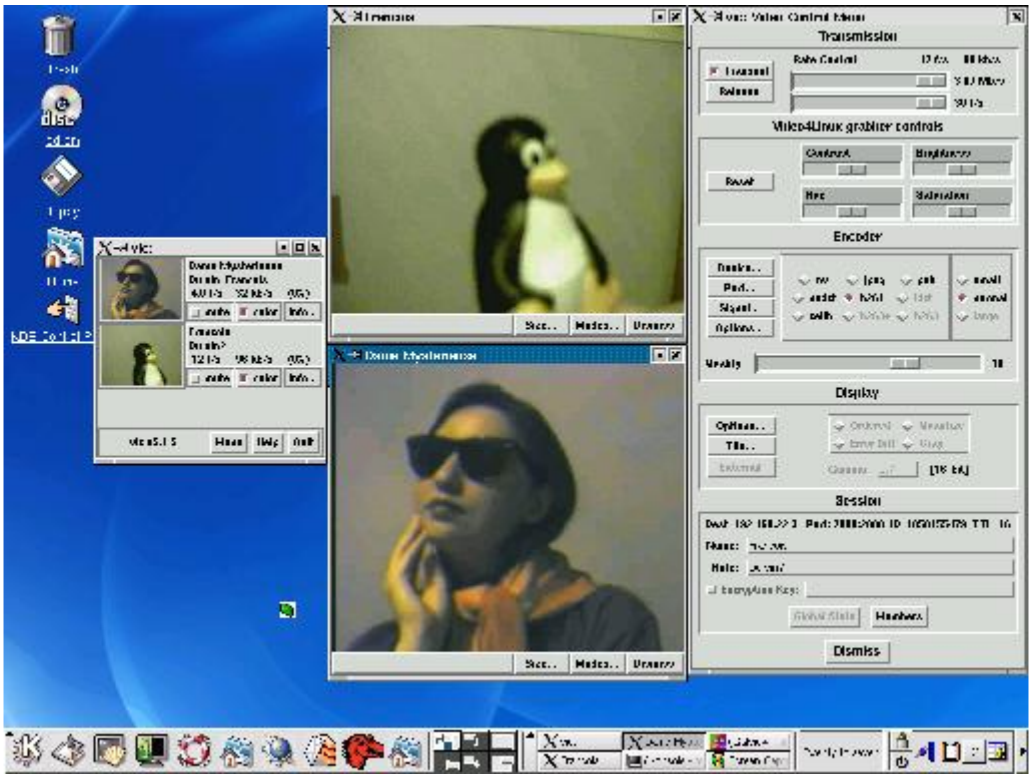


Figure 3. *Elle est une dame mystérieuse.*

Like the other applications we have seen today, vic's video control menu lets you modify a number of settings such as brightness, contrast and color. You even can change the size of the viewers by clicking on the Size button.

Once again, closing time approaches. Normally, I would ask my waiter to provide you with a final top-up, but I fear, *mes amis*, that it will be up to me to refill your glasses this evening. François seems somewhat preoccupied.

Ah, *l'amour*.

Resources

Marcel Gagné (mggagne@salmar.com) is president of Salmar Consulting, Inc., a systems integration and network consulting firm, and the author of Linux System Administration: A User's Guide, published by Addison-Wesley.

Verifying Filesystem Integrity with CVS

Michael Rash

Issue #94, February 2002

Even the most conscientious of paranoid penguins needs a vacation, so we have a guest columnist for Mick Bauer this month.

Paranoid Penguin

Verifying Filesystem Integrity with CVS

Even the most conscientious of paranoid penguins needs a vacation sometime, so we have a guest columnist for Mick Bauer this month.

by Michael Rash

The single most effective way to catch host-based intrusions is by detecting changes in the filesystem. Toward this end, Tripwire is the best known piece of software that automates the process of verifying filesystem integrity on a machine over time and alerting the administrator if any unauthorized changes are detected. The types of changes that Tripwire can detect include changes in file permissions and type, inode number, link count, uid and gid, size, access timestamp, modification timestamp, inode change timestamp and one-way hash signature (generated by any of ten different signature functions, including md5 and Snefru). In this article we explore the use of the Concurrent Versions System (CVS), together with some Perl glue, as a filesystem integrity checker in a manner similar to Tripwire. Although Tripwire is the best known filesystem integrity checker, it should be noted that Tripwire recently became a proprietary product to which there are many free and/or open-source alternatives, such as AIDE and FCheck (see Resources).

Tripwire

The basic process of deploying Tripwire on a system can be summed up by the following sequence of steps: 1) install the operating system, and immediately run Tripwire from the console before exposing the machine to any type of network connection; 2) store the resulting Tripwire database of filesystem information on read-only media separate from the system; and 3) periodically run Tripwire against the original database so a comparison can be made between the current filesystem information and the original database in order to look for any anomalies.

CVS

CVS is a tool used primarily by software developers to help organize and provide a version structure to a set of source code. One of its most useful features is the ability to keep track of all changes in a piece of source code (or ordinary text) over time, starting from when the code initially was checked into the CVS repository. When a developer wishes to make changes, the code is checked out of the repository, modifications are made and the resulting code is committed back into the repository. CVS automatically increments the version number and keeps track of the changes. After modifications have been committed to the repository, it is possible to see exactly what was changed in the new version relative to the previous (or any other) version by using the **cvs diff** command.

Now that we have a basic understanding of the steps used to administer Tripwire for a system, we will show how CVS can be leveraged in much the same way, with one important enhancement: difference tracking for plain-text configuration files. If one of your machines is cracked, and a user is added to the /etc/passwd file, Tripwire can report the fact that any number of file attributes have changed, such as the mtime or one-way hash signature, but it cannot tell you exactly which user was added.

A significant problem for intrusion detection in general is that the number of false positives generated tends to be very high, and host-based intrusion-detection systems (HBIDS) are not exempt to this phenomenon. Depending on the number of systems involved, and hence the resulting complexity, false positives can be generated so frequently that the data generated by HBIDS may become overwhelming for administrators to handle. Thus, in the example of a user being added to the /etc/passwd file, if the HBIDS could report exactly which user was added, it would help to determine whether the addition was part of a legitimate system administration action or something else. This could save hours of time since once it has been determined that a system has been compromised, the only way to guarantee returning the system to its normal state is to re-install the entire operating system from scratch.

Collecting the Data

In keeping with the Tripwire methodology of storing the initial filesystem snapshot on media separate from the system being monitored, we need a way to collect data from remote systems and archive it within a local CVS repository. To accomplish this, we set up a dedicated CVS “collector box” within our network. All filesystem monitoring functions will be executed by a single user from this machine. Monitoring functions include the collecting of raw data, synchronizing the data with the CVS repository and sending alerts if unauthorized changes are found. We will utilize the ssh protocol as the network communication vehicle to collect data from the remote machines. To make things easier we put our HBIDS user's RSA key into root's authorized_keys file on each of the target systems. This allows the HBIDS user to execute commands as root on any target system without having to supply a password via ssh. Now that we have a general idea for the architecture of the collector box, we can begin collecting the data.

We need to collect two classes of data from remote systems: ASCII configuration files and the output of commands. Collecting the output of commands is generally a broader category than collecting files because we probably are not going to want to replicate all remote filesystems in their entirety. Commands that produce important monitoring output include md5sum (generates md5 signatures for files) and find / -type f -perm +6000 (finds all files that have the uid and gid bits set in their permissions). In Listings 1 and 2, we illustrate Perl code that collects data from remote systems and monitors changes to this data over time.

Listing 1. collector.pl

In Listing 1, we have a piece of Perl code that makes use of the Net::SSH::Perl module to collect three sets of data from the host whose address is 192.168.10.5, md5 hash signatures of a few important system binaries, a tar archive of a few OS configuration files and a listing of all files that have the uid and/or gid permission bits set. Lines 7 and 8 define the IP address of the target machine as well as the remote user that collector.pl will log in to. Recall that we have the local user's preshared key on the remote machine, so we will not have to supply a password to log in. Lines 10-13 define a small sample list of system binaries for which md5 hash signatures will be calculated, and similarly, lines 15-18 define a list of files that will be archived locally from the remote system. Lines 20-24 build a small hash to link a local filename to the command that will be executed on the remote system, and the output of each command will be stored locally within this file. Lines 27-33 comprise the meat of the collection code and call the run_command() subroutine (lines 36-49) for each command in the %Cmds hash. Each execution of run_command() will create a new Net::SSH::Perl object that will be used to open an ssh session to the remote host, log in and execute the command that was passed to the subroutine.

Listing 2. cvschecker.pl

In Listing 2, we illustrate a piece of Perl code that is responsible for generating e-mail alerts if any of the files collected by collector.pl change. This is accomplished first by checking the current 192.168.10.5 module out of the CVS repository (line 12), executing collector.pl (line 14) to install fresh copies of the remote command output and files within the local directory structure, and then committing each (possibly modified) file back into the repository (line 20). By checking the return value of the cvs commit command for each file (line 20), we can determine if changes have been made to the file, as cvs automatically increments the file's version number and keeps track of *exactly* what has changed. If a change is detected in a particular file, cvschecker.pl calculates the previous version number (lines 27-36), executes the cvs diff command against the previous revision to get the changes (lines 39-40) and e-mails the contents of the change (lines 47-48) to the e-mail address defined in line 7.

Detecting Intrusions

Now let's put the collector.pl and cvschecker.pl scripts into action with a couple of intrusion examples. Assume the target system is a Red Hat 6.2 machine; HIBDS data has been collected from this machine before any external network connection was established, and the target has an IP address of 192.168.10.5.

Example 1

Suppose machine 192.168.10.5 is cracked, and the following command is executed as root:

```
# cp /bin/sh /dev/... && chmod 4755 /dev/...
```

This will copy the /bin/sh shell to the /dev/ directory as the file “...” and will set the uid bit. Because the file is owned by root, and we made it executable by any user on the system, the attacker only needs to know the path /dev/... to execute any command as root. Obviously, we would like to know if something like this has happened on the 192.168.10.5 system. Now, on the collector box, we execute **cvschecker.pl**, and the following e-mail is sent to root@localhost, which clearly shows /dev/... as a new suid file:

```
From: hbids@localhost
Subject: Changed file on 192.168.10.5: suidfiles
To: root@localhost
Date: Sat, 10 Nov 2001 17:35:13 -0500 (EST)
Index: /home/mbr/192.168.10.5/suidfiles
===================================================
RCS file: /usr/local/hbids_cvs/192.168.10.5/suidfiles,v
retrieving revision 1.3
retrieving revision 1.4
diff -r1.3 -r1.4
4a5
> -rwsr-xr-x 1 root root 512668 Nov 10 18:40 /dev/...
```

Example 2

Now suppose an attacker is able to execute the following two commands as root:

```
# echo "eviluser:x:0:0::/:/bin/bash" >> /etc/passwd
# echo "eviluser::11636:0:99999:7:::" >> /etc/shadow
```

Note that the uid and gid for eviluser are set to 0 and 0 in the /etc/passwd entry, and also that there is no encrypted password string in the /etc/shadow entry. Hence, any user on the system could become root without supplying a password simply by typing **su - eviluser**. As in the previous example, after running **cvschecker.pl**, we receive the following e-mails in root's mailbox:

```
From: hbids@localhost
Subject: Changed file on 192.168.10.5: /etc/passwd
Delivered-To: root@localhost
Date: Sat, 10 Nov 2001 17:43:17 -0500 (EST)
Index: /home/mbr/192.168.10.5/etc/passwd
=======================================================
RCS file: /usr/local/hbids_cvs/192.168.10.5/etc/passwd,v
retrieving revision 1.2
retrieving revision 1.3
diff -r1.2 -r1.3
26a27
> eviluser:x:0:0::/:/bin/bash
```

and

```
From: hbids@localhost
Subject: Changed file on 192.168.10.5: /etc/shadow
Delivered-To: root@localhost
Date: Sat, 10 Nov 2001 17:43:18 -0500 (EST)
Index: /home/mbr/192.168.10.5/etc/shadow
=======================================================
RCS file: /usr/local/hbids_cvs/192.168.10.5/etc/shadow,v
retrieving revision 1.2
retrieving revision 1.3
diff -r1.2 -r1.3
26a27
> eviluser::11636:0:99999:7:::
```

Conclusion

Finding changes in the filesystem can be an effective method for detecting intruders. In this article we have illustrated some simple Perl code that bends CVS into a homegrown, host-based intrusion-detection system. At my current place of employment, USinternetworking, Inc., a large ASP in Annapolis, Maryland, we use a similar (although greatly expanded) custom system called USiOasis to help verify filesystem integrity across several hundred machines in our network infrastructure. The machines are loaded with various operating systems that include Linux, HPUX, Solaris and Windows, and run many different types of server applications. The system includes a MySQL database back end, a rather large CVS repository and a custom web/CGI front end written mostly in Perl. Making use of a CVS repository to perform difference tracking also comes with an important additional benefit: an excellent visualization tool written in

Python called ViewCVS. Storing operating system and application configuration files within CVS also aids several areas outside of detecting intrusions, such as troubleshooting network and application-level outages, disaster recovery and tracking system configurations over time.

Resources

Michael Rash (mbr@cipherdyne.com) works as a senior security engineer for an ASP in Annapolis, Maryland. He holds a Master's degree in Applied Mathematics from the University of Maryland and has been tinkering with Linux since 1998. In his free time he enjoys playing the violin for the Prince George's Philharmonic Orchestra.

Archive Index Issue Table of Contents

Advanced search

NLE Video Editors

Robin Rowe

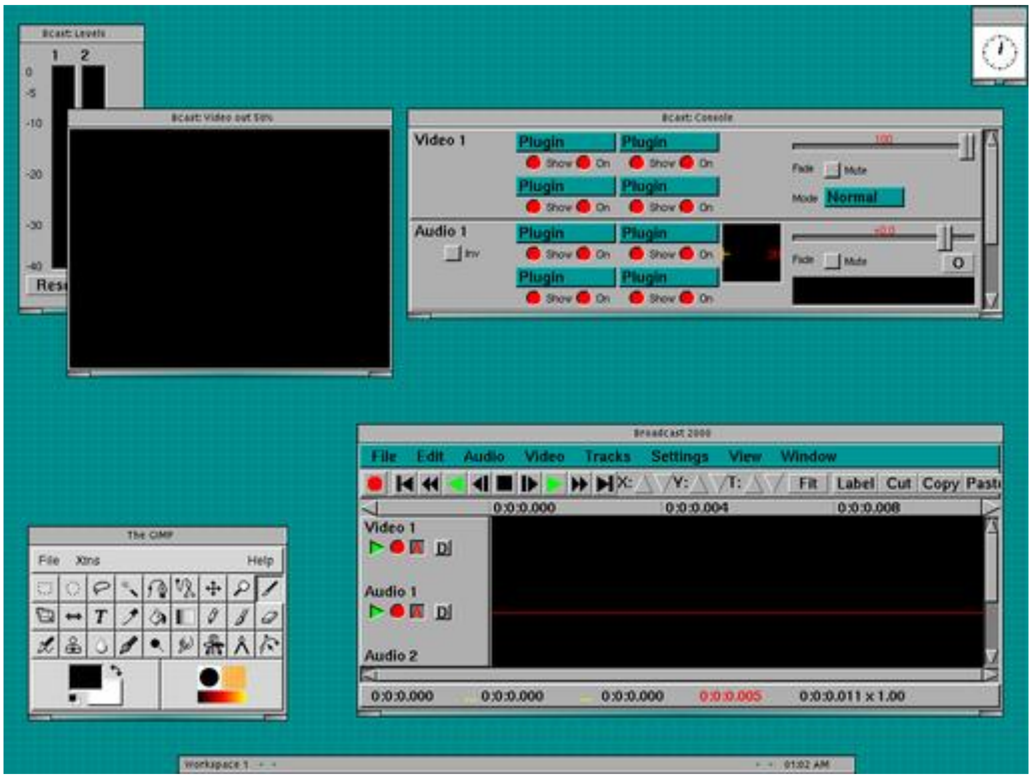
Issue #94, February 2002

Robin takes a look at six Linux video nonlinear editing (NLE) sofware packages: Broadcast 2000, Crow, Kino, LVE, MainActor and Trinity.

In alphabetical order, we evaluate Broadcast 2000, Crow, Kino, LVE, MainActor and Trinity. Except for MainActor, all are open-source software. Especially intriguing are Kino, a highly regarded DV editor that just released a new version, and LVE, a little-known German-made MPEG editor.

Broadcast 2000

Broadcast 2000 has been removed by the developer from its main web site, making some think that it is no longer exists, but video systems integrator Linux Media Arts (LMA) still offers it. LMA President Mike Collins says, “We want to continue development of Broadcast 2000, both as a free, open-source project and as bundled commercial software with provided support.” Collins says he is trying to arrange putting Broadcast 2000 on SourceForge.net, but until then, anyone may contact him directly about downloading it. Broadcast 2000 is available as an RPM or a source tarball.



The Broadcast 2000 video editor, based on QuickTime, offers many advanced features including effects using the GIMP.

Broadcast 2000 was featured in the January 2001 issue of *Linux Journal* [see Adam Williams' article “Moviemaking on a Linux Box? No Way!”]. Our installation is a little different because our distribution isn't Red Hat, but Debian. Let's convert the Red Hat-compatible RPM into a Debian deb (Listing 1).

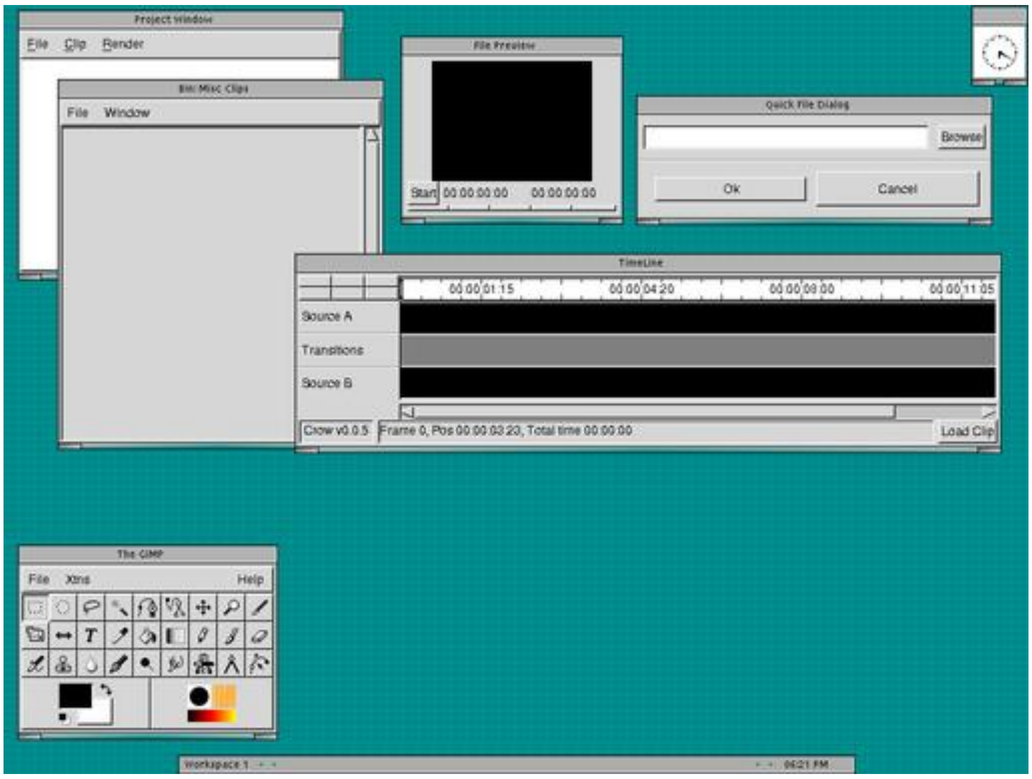
Listing 1. Converting the Red Hat-Compatible RPM into a Debian deb

Alien works well at converting RPM archives into deb format, but it doesn't find shared libraries that are in nonstandard locations (note the warning from Alien that it can't find libbcbase.so). You will need to remember that after installing the deb with dpkg. Once Broadcast 2000 is installed, search for the lost DLL and add its directory path to your ld.so.conf. Then run **ldconfig** to update the runtime linker settings. Otherwise, the program won't run because it won't be able to find its shared libraries.

Broadcast 2000 record formats include WAV, PCM, QuickTime and JPEG image sequence. The supported QuickTime (MOV) types are JPEG photo, motion JPEG A, PNG, PNG with alpha, uncompressed RGB, uncompressed RGBA, YUV 4:2:0 planar and YUV 4:2:2 packed. That's a lot of formats, but they aren't as compatible with other tools and operating systems as you might expect.

Crow

Crow is a project by Eric Fry in Australia. Although only an alpha version, we still were curious to try Crow. It is only available for download via the CVS version control system.



Crow's developer says it is too incomplete to use, but we learned a lot about configuring and compiling alpha projects in the process of building it.

Our Linux box is on a Windows network behind a Windows firewall with Cox cable modem. We would have to configure our firewall to pass CVS. Using Telnet we checked that we could reach SourceForge.net CVS port 2401 manually. We then created a 2401 TCP wrapper in WinGate on our SOHO firewall machine (named gap). Our TCP wrapper was set to point to cvs.Crow.sourceforge.net.

Attempting a CVS login from my Linux box (tbird) failed—something about .cvspass not being configured correctly on that machine. However, we were able to do a CVS checkout. That was all we actually needed.

```
cvs -z3 -d:pserver:anonymous@gap:/cvsroot/crow co crow
```

If we hadn't been drilling through our firewall, the CVS command would have specified cvs.Crow.sourceforge.net rather than gap.

Crow objected that it had the wrong libtool ltconfig version, but we fixed that with libtoolize:

```
libtoolize --force
aclocal
./configure
make
```

No dice—the Crow make failed because an include file couldn't find gtk/gtk.h. The Makefiles didn't have the paths configured in a generic way to point to the GTK++ GUI headers, and our GTK++ files were installed in a different directory. This called for a little autoconf surgery. We added GNOME_INIT to configure.in so it would emit the GTK_CFLAGS and GTK_LIBS variables. Then we substituted those variables, e.g., $(GTK_CFLAGS), in 14 different Makefile.am files used by Crow. After regenerating the configure file, Crow built. Bingo, we got it!

```
aclocal
autoconf
./configure
make
ulimit -c unlimited
./app/crow
```

We didn't do a **make install** of Crow because the INSTALL file had warned us not to. We set ulimit to be able to generate core files if we had a crash. And, when we ran Crow it immediately crashed with a segmentation fault. Let's debug that using the gdb debugger:

```
gdb app/.libs/lt-crow core
> bt
```

Because ./app/crow is a script, we had to figure out the actual name of the executable to load in gdb. Running a backtrace (**bt**) to examine the function stack didn't do us any good at this point because there was no debug information in our build. We first would have to rebuild with debug enabled:

```
make clean
make -e ?CC=gcc -g?
```

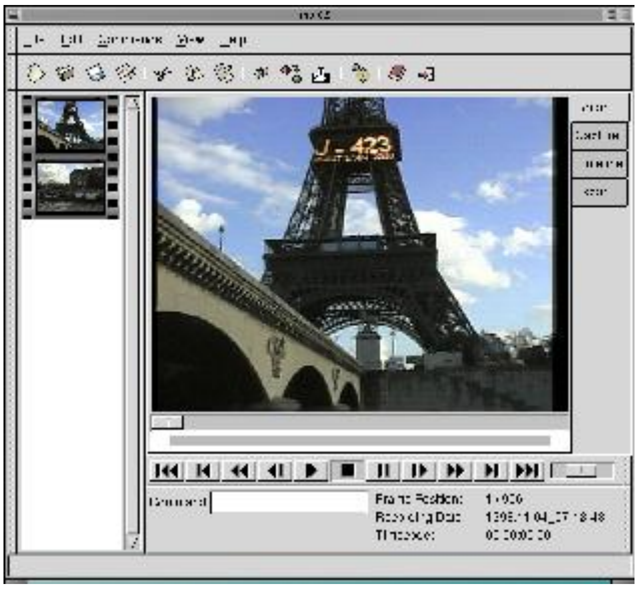
Now we could load gdb again and see the relevant call stack and source code. That immediately revealed that the crash was due to a null pointer variable dir in app/plug_in.c on line 275. We simply inserted a line there to return from the function before using the variable if it was 0:

```
if(!dir) return 0;
```

Crow is very much an alpha and too incomplete for use. Fry says that his files on SourceForge.net are an old version and that he is planning to rewrite it. He wants to talk with others interested in designing an editing application for Linux from the ground up. “I'd like to have a fully featured editor capable of doing real-world editing tasks like 3:2 pull-down, EDL output database clip management and HD playback”, says Fry.

Kino

Kino is a simple cuts-only DV editor. Kino just released version 0.50, a major new release. To install Kino, you first must install many drivers. That, and installing some of the many other DV-related programs available, makes for a rather involved process.



Kino, a popular Linux editor for DV, just released a new version.

The instructions recommend using the latest kernel to get the best IEEE 1394 FireWire DV support. Using **make xconfig**, we were puzzled at first that the IEEE 1394 options were grayed out and couldn't be selected. In configuring our 2.4.14 kernel, we first had to select the code maturity menu option to enable the 1394 menu.

The following drivers were next: libavc1394-0.3.1.tar.gz, libdc1394-0.8.3.tar.gz, libraw1394_0.9.0.tar.gz, libdv-devel-0.9-1.i386.rpm and libdv-0.9-1.i386.rpm.

For tarballs, let's follow the standard procedure:

```
tar xvfz libdc1394-0.8.3.tar.gz
cd libdc1394-0.8.3
./configure
make
su
make install
```

Those that came as RPMs we installed using Alien. If you build libdv from the tarball you will need to install pkg-config first to run configure successfully.

The libraries include some simple tools. Included with libraw1394 is testlibraw. Let's start with that to see if it detects our FireWire cards. We have two FireWire PCI cards in our Athlon PC, a PYRO card and a no-name FireWire card included with our FireWire scanner. Both are OHCI-compliant, and detect properly.

However, we had to **chmod** the device to gain permissions access for ordinary users.

```
chmod 666 /dev/raw1394
./src/testlibraw
  successfully got handle
  current generation number: 3
  2 card(s) found
    nodes on bus:  1, card name: ohci1394
    nodes on bus:  1, card name: ohci1394
  using first card found: 1 nodes on bus,
    local ID is 0, IRM is 63
  ,
  ,
  ,
```

The libavc1394 romtest program enumerates the ROM information of any attached FireWire device. It correctly detects our Sony TRV8 DV camcorder.

Included with libavc1394 is dvcont, a remote control for DV camcorders. When using dvcont the first time, specify **dvcont help**, so you know what the commands are. If dvcont can't find its drivers, it exits immediately and won't even display help. You may need to **modprobe** video1394 and raw1394. With no command specified, dvcont does nothing. With your camcorder in play mode, executing **dvcont stop** causes the camcorder to stop. There are similar commands for the other camcorder functions.

Listing 2. modprobe

Included with libdv are playdv and encodedv, but both died with segmentation faults. These are supposed to play or encode a .dv file.

Here are some of the many available DV programs that may be useful: dvgrab-1.01.tar.gz, gscanbus-0.6.tgz and gstreamer-0.2.1.tar.gz.

We couldn't build coriander or gscanbus. **gstreamer** built, although it took a long time and then died with a segmentation fault. The tool we really wanted from this set was dvgrab, so we could do a console test of DV capture. Since it didn't prevent us from using Kino, we didn't stop to investigate why we had problems with some of the DV tools.

The dvgrab utility will copy a video playing on your camcorder to your PC. It has no device control, which means if the camcorder isn't in play mode, dvgrab simply waits. If no FireWire DV stream is detected it won't do anything.

```
dvgrab --frames 30 test
ls -l test*
  -rw-r--r--    1 rower    rower
  2471424 Nov 17 17:27 test001.avi
```

We grabbed a second of video as a test. It was saved by dvgrab as a DV stream wrapped in the AVI transport. Arne Schirmacher created dvgrab and Kino.

We initially built the 0.46 version of Kino (kino-0.46.tar.gz), but 0.5 was released during our evaluation (kino-0.5.tar.gz). At first we had a problem building Kino because **configure** couldn't find gnome-config. That seemed odd since we had GNOME installed (on Debian Sid). Doing a search for it in dpkg should report gnome-config in libgnome-dev (as shown below), but we came up empty.

```
dpkg -S gnome-config
  libgnome-dev: /usr/include/libgnome/gnome-config.h
  libgnome-dev: /usr/bin/gnome-config
  libgnome-dev: /usr/share/man/man1/gnome-config.1.gz
```

Kino developer Dan Dennedy suggested that we install Ximian GNOME because that is his configuration. So we added a link to our sources.list and installed it:

```
vi /etc/apt/sources.list
deb http://red-carpet.ximian.com/debian stable main
.
.
.
apt-get update
apt-get install task-ximian-gnome
```

Installing Ximian created massive conflicts with already-installed GNOME components. We had to uninstall many packages that were reported broken by apt-get. It was actually a lot more effort, but the example shown here represents the gist of the process. Surprisingly, it was the GNOME GTK docs that seemed to have our install the most wedged. Eventually we got Ximian installed as well as the Ximian GNOME development libraries. Kino built without a problem.

```
dpkg -r libgnome-dev
apt-get build-dep libgtk1.2-dev
dpkg -r libgtk1.2-doc
apt-get -f install
apt-get install libxml2
```

Kino is based on a metaphor of the vi command set. Commands for lines, words and characters become movies, clips and frames. Use **d0** to set an in-point and **d$** to set an out-point.

Dennedy says, “The new version of Kino is a major rewrite, with new user interface, no multiple-floating windows and a storyboard view more like iMovie. You'll notice greater consistency in interface.” Dennedy says the Kino team has a new developer, Charles Yates, who was the major force on version 0.5.

Dennedy notes that the Kino EDL is in XML SMIL format, and that sound support is based on the OSS interface. Because most sound cards don't support multiple opens, esound is only supported with the SBLive! card. The Contour ShuttlePro USB controller ($125) provides a convenient device to control Kino, bypassing the vi-based interface. Kino supports XVideo for better preview performance, but we had to turn that feature off because the preview window became stuck in place permanently on our desktop. That seems to be due to our having installed the wrong version of the ati.2 driver, and not a problem with XVideo generally.

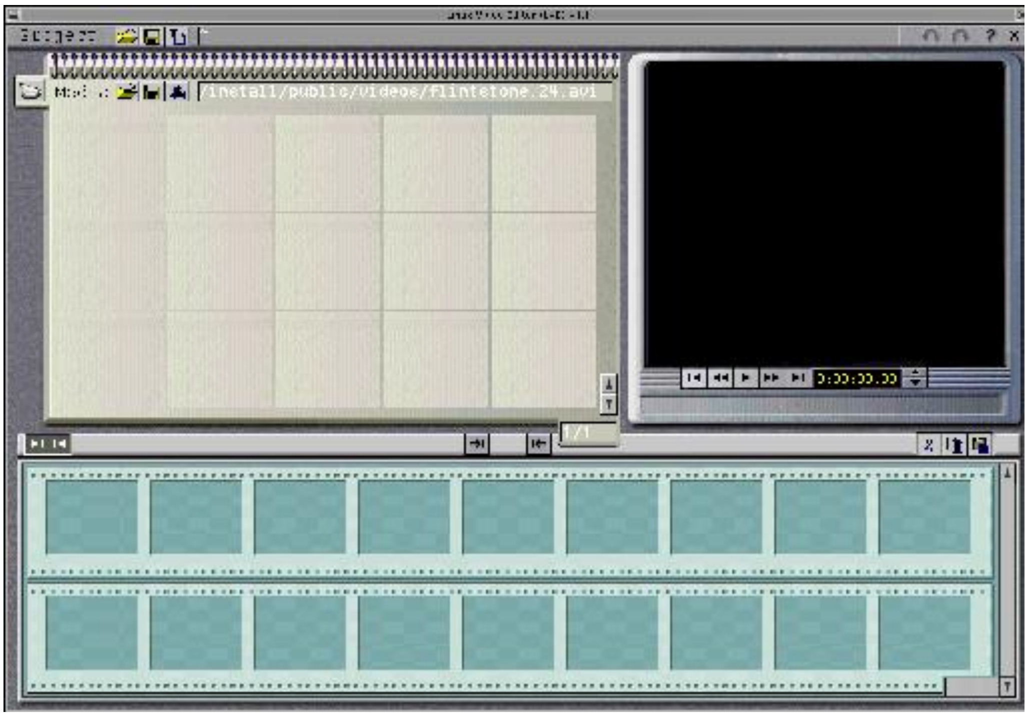
Kino can handle DV1 or DV2 AVI files, but not raw DV files (.dv). The format you choose affects compatibility with other tools. Dennedy explains, “Windows Media Player (WMP) and Linux Avifile will play DV2 AVI format, provided the Microsoft DLL is installed. That's called qdve.dll. MPlayer also can play those but requires adding a line to its configuration file.” Dennedy notes that WMP also can handle DV1 AVI. He says MainActor can handle DV but has no DV camcorder I/O.

Tools such as dv2jpeg can help you move between formats. Dennedy adds, “mjpegtools can read DV AVI to create a VCD; mjpegtools has support for MPEG-1 VCD and MPEG-2 SVCD. Kino will soon export MPEG-1 and MPEG-4 formats based on FFmpeg, but FFmpeg is currently lacking MPEG-2.” And, mjpegtools can split off the DV audio. The tool transcode will convert DV AVI to OpenDivX.

Dennedy is excited that Kino now supports DV export back to the camcorder. He says Kino is the only DV application to offer that besides the libdv console program. Kino doesn't offer a logging mode, and Dennedy doesn't expect it will soon. He says that feature isn't as interesting to the team as adding effects and that a new dv1394 driver is in development that will provide scrub preview in the TV monitor. He also says Kino MPEG export is coming but has no plans to integrate with Film GIMP.

LVE

Linux Video Editor (LVE) developer Gerhard Monzel says, “The intended purpose was cutting MPEGs and to encode to (S)VCDs.” Monzel works at SAP as systems administrator in St. Ingbert, Germany. The LVE documentation is all in German. “LVE is based on some freeware libraries: libmpeg3 to seek and decode MPEG and libsdl as base of my GUI”, says Monzel. “The rest is self-made.” He says many MPEG formats are supported (MPEG-1 and MPEG-2) including VOB and IFO, but not DVB. His GUI design was influenced by Pinnacle Studio MP10.

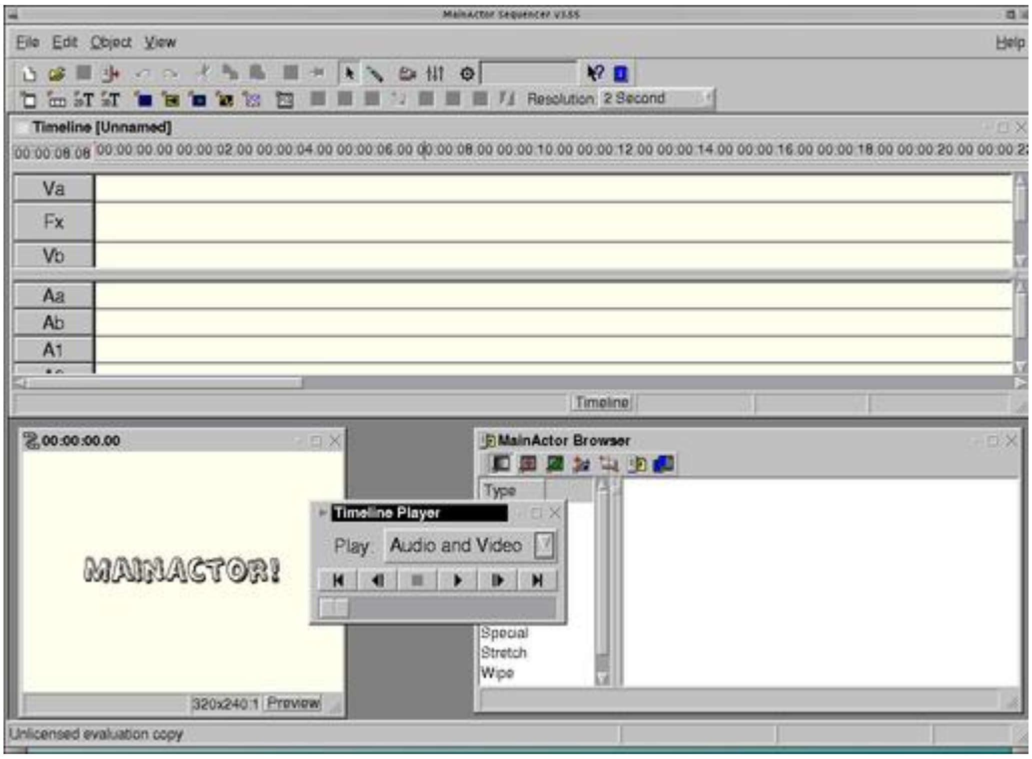

LVE is a German-made MPEG editor intended for making (S)VCDs.

LVE is cuts-only—no effects and no titles. Note that whether the source material is PAL or NTSC, the output is always PAL. NTSC sources must be processed with sox to correct the framerate changing the audio pitch. LVE has no install procedure. The tarball must be extracted in the root directory:

```
cd /
tar xvfz /install/public/nle/lve/
lve_bin-31-10-01.tar.gz
ls /usr/local/lve/bin
  bbainfo  bbinfo   bbvinfo  ffmpeg_lve
  gensmart  lmp  mplex  toolame
  bbdmux   bbmplex  encode   genmpg
  gensvcd   lve  qdir
chmod 666 /usr/local/lve/lib/SystemFont.bmp
./lve
```

MainActor

MainActor is the only closed-source application we looked at. We installed it from RPM using Alien, following much the same procedure as with Broadcast 2000.

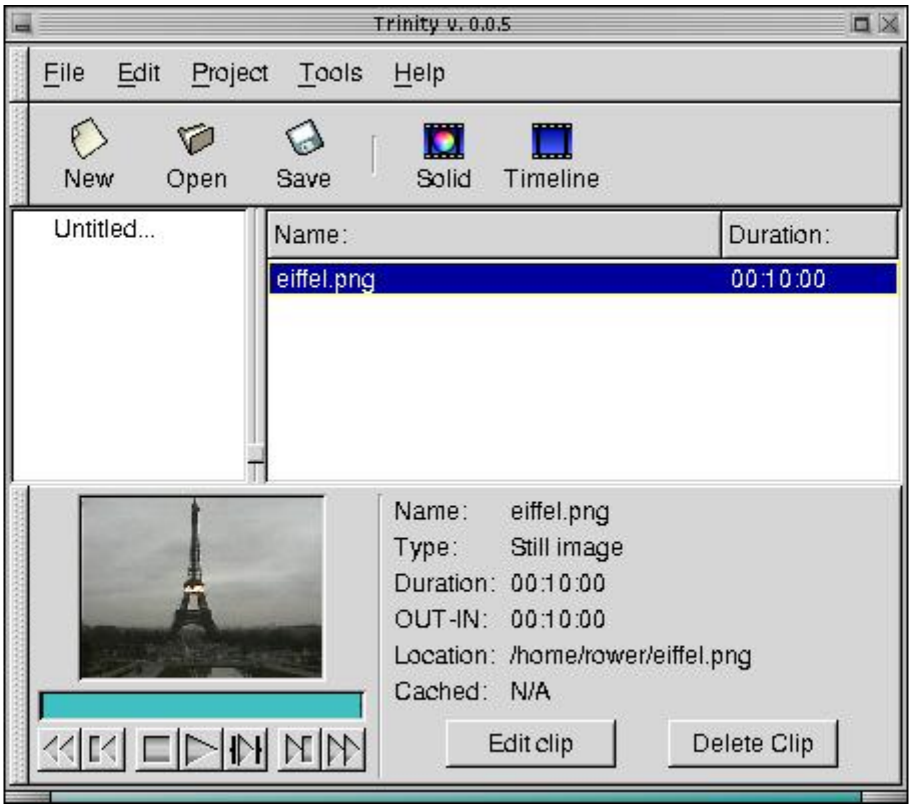


MainActor is a shareware editor with Linux and Windows versions.

MainActor can title (2-D and 3-D text) and edit movies. It offers video transitions and audio effects. The evaluation version writes “MainActor” on your video until you purchase a registered copy. MainActor includes the following programs: maseq (NLE), mave (animation converter), macap (V4L MJPEG capture) and lmatool (console video file converter). The documentation is in /usr/share/doc/Packages/MainActor.

Trinity

Trinity editor and maintainer Chris Hardy says it has rudimentary MPEG and audio support. Trinity can understand sequences of image frame files, too. “The source code hasn't been touched in two years”, says Hardy. “I haven't heard from the developer in a while and the project has been in limbo.” The GUI is one thing Hardy likes about Trinity.



Name:	Duration:
eiffel.png	00:10:00

Name: eiffel.png
Type: Still image
Duration: 00:10:00
OUT-IN: 00:10:00
Location: /home/rower/eiffel.png
Cached: N/A

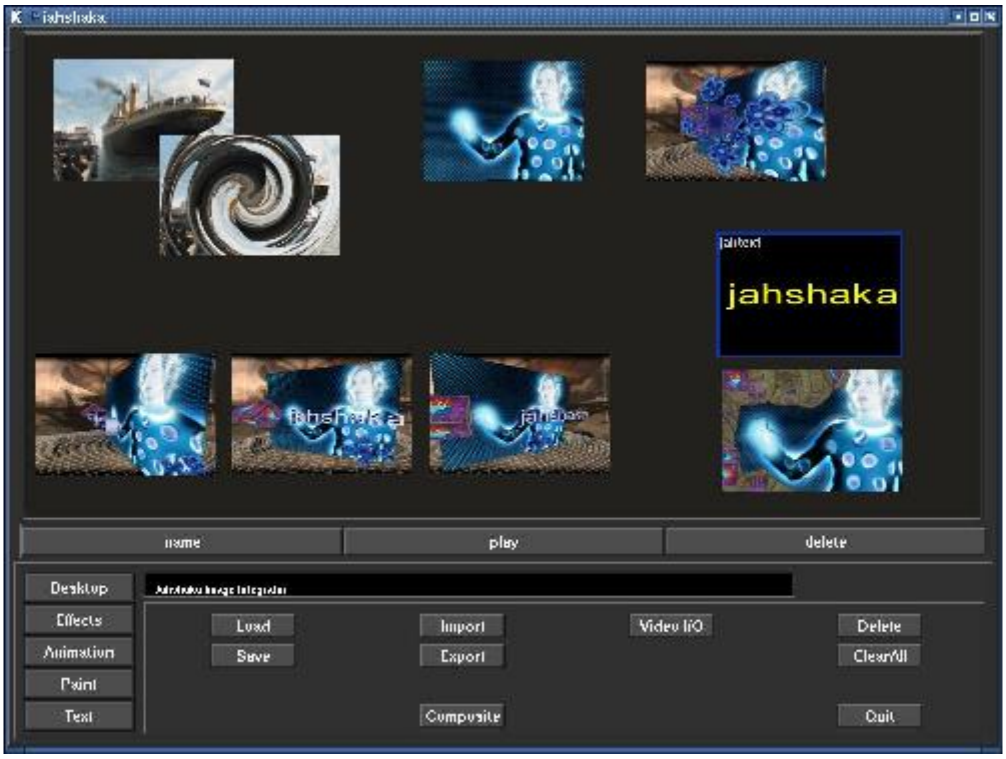
Trinity is in search of a developer to lead and rename this project.

There were several minor compilation errors that we fixed while building Trinity 0.5 and one serious one. We sent the corrections to Hardy. Because of a conflict with a commercial product, Trinity must be renamed, and Hardy would like a developer interested in working on an NLE to join the project so it can go forward.

Conclusion

Computers present a great advantage in editing movies because they can easily edit out of sequence or remove or add a scene—that's why it's called nonlinear editing. Nonlinear editors are used to edit video sequences to create television shows and motion pictures.

In examining Broadcast 2000, Crow, Kino, LVE, MainActor and Trinity, we've looked at some of the video tools available in Linux, but hardly all of them. FFmpeg, GAnSO, Gnonlin, Jahshaka, Linux Video Studio, matterial, mpgtx, mpegcut and SAMPEG-2 present more choices.



Jahshaka, a special effects compositor, has both Linux and Windows versions based on OpenGL.

Two commercial (expensive) tools that we will be evaluating in the future are Nothing Real Shake and Silicon Grail RAYZ. These are used to add special effects to many Hollywood motion pictures. There is just one major open-source tool that is used in major motion pictures and that is Film GIMP, used in *Harry Potter*, *Cats & Dogs*, *Lord of the Rings* and more. We'll take a look at Film GIMP next time.

Resources

email: Robin.Rowe@MovieEditor.com

Robin Rowe (robin.rowe@movieeditor.com) is a partner in MovieEditor.com, a technology company that creates internet and broadcast video applications. He has written for *Dr. Dobb's Journal*, the *C++ Report*, the *C/C++ Users Journal* and *Data Based Advisor*.

On Irresponsible ISPs

David A. Bandel

Issue #94, February 2002

This month David covers everything from spam to calculating your body mass.

Focus on Software

On Irresponsible ISPs

This month David covers everything from spam to calculating your body mass.

by David A. Bandel

Just thought I'd touch on a subject near and dear to my heart this month: irresponsible ISPs—seems there's a number of them out there. I see them every day in my server logs and e-mail. Not long ago, with the court case in Australia, I found it necessary to remove my RBL entry in sendmail—a mistake. I generally receive around 200 e-mails per day. After removing the RBL entry, I received so many spams it drowned my e-mail. I had a choice: delete everything or waste all day deleting spam as I replied to the few legitimate e-mails I received. So if any of you have e-mailed me and not received a reply, try again. I'm methodically blocking the IPs of every spam I receive. I've already blocked 90% of Korea and Japan. I also have had to block what I understand is 80% of Germany, but for another reason: massive attacks on my servers by script kiddies and the intransigence of <@url>t-online.net (<@url>t-online.de) to do anything about them. I'm getting ready to do the same to wanadoo.fr. I almost blocked sympatico.ca, but when I told them I was about to block them, they took action. It shouldn't have to be this way. If I had time, I'd put up an SQL server with a list of IPs/IP blocks/domains that condone criminal behavior (spam relays and attacks on servers). Both steal resources that I (and you) pay for. Don't expect legislation to help until lawmakers begin to depend on e-mail and are buried in spam. Until then, my blocked list grows by the day. Too bad for legitimate users.

fwmon www.sourceforge.net/projects/firestorm-ids

For those of you firewall buffs out there, this utility should interest you. The biggest problem with using it is that it requires you to compile iptables with Rusty Russell's patch-o-matic because according to the documentation, the iptables NETLINK target is forever consigned to experimental status. You'll need the kernel netlink device module also. After that, you can craft a nonterminating rule for any filter table chain with the target -j NETLINK. Then run **fwmon**. Watching packets flow on the screen is amazing. And because it writes to stdout, you can pipe it to any program for almost any purpose. Requires: iptables w/NETLINK, kernel compiled with NETLINK device, glibc.

ConnMon www.student.lu.se/~nbi98oli

This utility will do connection monitoring similar to netwatch. However, with a kernel patch, you can have statistical tracking (bandwidth usage) monitoring with it. ConnMon uses the adns package, which is an extremely fast set of asynchronous resolver libraries and tools. Requires: libadns, libncurses, glibc.

BMI ibiblio.org/bmi

If you want to know your body mass index, this little utility will calculate it for you. If you're used to the US standards of height and weight in feet and inches and pounds, you'll want to make sure you have the units utility to convert. I'd note that this is body mass (not body fat) because it doesn't take into account muscle mass (which is denser) vs. fat. So you'll need a little more information than simply the numbers presented, but if you know what your target body mass index is, this can help you monitor your results. Requires: libstdc++, libm, glibc.

Rotix elektron.its.tudelft.nl/~hemmin98/rotix.html

For those of you who remember (and perhaps even used) ROT-13, and noticed that it's missing from most, if not all modern distros, Rotix will make a nice replacement. By default, Rotix performs rot-13 rotation but can be told to perform other rotations as well. Based on some of the mailing-list postings I've seen, Rotix would be a good addition to the repertoire of a number of posters. Requires: glibc.

Noguska On-Line Accounting System nola.noguska.com

If you're a medium-sized business with complex accounting needs, this may be something for you. Small businesses and SOHOs may do better with SQL-Ledger, but NOLA provides a number of advantages. NOLA offers everything from point of sale, to inventory and more. Presently, it is available only in

English and Portuguese, but expect that to expand soon. My only note about this is when you're ready to try to install it, all the documentation is written in MS Word—not exactly an application available on Linux servers for reading installation documents. Requires: web server with PHP and MySQL, MySQL server, JavaScript-capable web browser.

Calcoo calcoo.sourceforge.net

Calcoo is a very easy-to-read and use scientific calculator. You also can choose between standard and RPN (reverse polish notation) for doing the calculations. About the only thing this calculator lacks is the ability to switch from decimal to hex to octal to binary. Requires: libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, glibc.

Ministry of Truth mot.sourceforge.net

I looked at several apps from three years ago, and while some of you may have liked GABY or Gentoo better, this is my pick. I can't say that it's changed a lot, but then, I didn't see a need for it to change. I continue to use this little jewel. What more needs be said? Track hardware, software, users and jobs. While it appears to be in maintainer mode, it works and works well. Requires: Apache, PHP, MySQL. Until next month.

email: david@pananix.com

David A. Bandel (dbandel@pananix.com) is a Linux/UNIX consultant currently living in the Republic of Panama. He is coauthor of Que Special Edition: Using Caldera OpenLinux.

Embedded Linux and Java—Wave of the Future?

Rick Lehrbaum

Issue #94, February 2002

Rick contemplates the coming exponential proliferation of smart devices using Java + Linux.

Focus on Embedded Systems

Embedded Linux and Java—Wave of the Future?

Rick contemplates the coming exponential proliferation of smart devices using Java + Linux.

by Rick Lehrbaum

The world of intelligent devices is changing dramatically. The computerized devices around us are getting smarter; they're increasingly connected and interdependent, and they're becoming vastly more numerous. And, all this is happening at an ever-increasing rate.

Blame it on Moore's Law, but it's now practical to embed moderately high-performance computing and connectivity in just about everything that runs on electricity—whether tethered or mobile. This trend is fueled by powerful and highly integrated system-on-chip processors, coupled with large-capacity system and storage memories (both disk and silicon), and empowered by wired and wireless communications interfaces (Ethernet, IrDA, 802.11, Bluetooth).

Another important phenomenon is that as both embedded computing and connectivity proliferate, the intelligence within tomorrow's devices is becoming less localized. Increasingly, the precise location of the software running on these devices is blurring, and eventually we're unlikely actually to know where the programs we use are located. Is the application running inside the device? Is it based on some remote server (e.g., a home services gateway)? Is it located at an internet-based application service provider? Is it a combination of all three?

Call it distributed intelligence or distributed computing. Call it .NET. Call it the post-PC era. Whatever you call it, one thing's certain: the era of isolated, autonomous desktop PCs running nothing but localized software is coming to an end, like the mammoths of the Ice Age.

Welcome to the Post-PC Era

As the boundaries of the traditional computing paradigm blur and a new reality based on distributed, interconnected, pervasive computing devices dawns, a few important attributes of the coming era draw into focus:

- The number of smart devices (i.e., products with embedded operating systems inside) will grow exponentially, reaching numbers in the billions.
- The choice of CPU will be more a matter of cost than technology or architecture.
- Nearly all devices will have connectivity, whether wired or wireless.
- Most devices will have the ability to be upgraded or repaired remotely, by downloading new firmware or software.
- Most devices will have specific rather than general-purpose functionality, so their application software will be defined by their manufacturers (rather than loaded by their users).

In general, most computing devices in this new era will *not* be PCs. Instead, they'll be smart appliances of various shapes and capabilities, used for information, entertainment, control and other purposes. Think of things like smart wristwatches (with built-in mobile phone and PIM functions), advanced cell phone/PDAs, audio/video systems, security systems, automobile infotronics, smart kitchen appliances and PC-like desktop terminals. The list goes on and on.

A Fertile Field for Linux

As the number and variety of devices with embedded intelligence grow exponentially, the need to minimize cost and maximize specialization increases correspondingly. Hence, embedded Linux becomes a highly desirable technology for the operating system due to its scalability, configurability and affordability.

It's worth noting that until recently, the cost penalty associated with the CPU and memory resources necessary to run Linux had been a somewhat limiting factor, relative to using it in cost-sensitive devices. Now, however, the baseline needs of embedding Linux—roughly 2MB Flash and 4MB RAM memory and a moderate speed processor—have become reasonably inexpensive, thanks in large measure to Moore's Law.

And Java

Another important challenge in this new era in which we'll be surrounded by billions of increasingly intelligent devices, all communicating with one another, is the obvious need to simplify and quicken the process of application development, deployment and maintenance. In this regard, Java appears postured to play an increasingly significant role.

Although Java failed to hit the target for which it was initially developed (which was, ironically, to serve as an embedded operating system within smart devices), Java ended up providing a convenient means to enable moving applications around among computing devices—propelled to this position by the dramatic emergence of the Web.

Today, despite its early failure as an embedded operating system, Java is showing promise in the role of providing a device-independent application platform, running on top of the embedded operating system. In this case, rather than serving as the operating system itself, Java provides the benefit of masking the unique aspects of the underlying device and providing an array of services beyond those offered by the embedded OS.

In the context of an exponential proliferation of smart devices, Java is emerging as a handy way to minimize device-specific development and to allow developers to focus on the truly unique aspects of their projects. Increasingly, Java is providing a means to obtain functionality like GUIs, web browsers, protocol stacks, handwriting and speech recognition, wireless communications, multimedia support, database management and a wide range of remote services.

A Couple of High-Profile Examples

One interesting example of a product based on a combination of embedded Java + Linux is a consumer entertainment system that was recently announced by Hewlett-Packard. The HP Digital Entertainment Center is basically a home-entertainment appliance that brings digital music and information via broadband and home networks into the living room, without a PC. The system connects like an audio component to a normal home stereo system and can be used to burn custom CDs, create/store/organize MP3 files on its large internal hard disk, transfer music to digital music players and listen to internet radio.

The HP Digital Entertainment Center

“HP embraced Linux for consumer appliances because of its open-source and community support”, explains William Woo, general manager of Hewlett-Packard's Embedded Software Operation. He adds:

> We have customized Linux for use as an HP embedded OS and combined it with our HP Chai technology to create an embedded software solution with a Java application environment and web connectivity. HP Chai supports Java applications for delivering e-services to create a rich consumer experience.

Another high-profile, Linux-based device that makes a point of its support for Java applications, of course, is Sharp's new Zaurus PDA. “The Zaurus SL-5000D offers incredible potential for the developer community”, said Steve Petix, associate vice president of Sharp's Mobile & IT Solutions Group. “We are excited to support both Linux and Java developers as they create next-generation mobile applications for this powerful new platform.”

Wave of the Future?

In the exciting new post-PC world of pervasive computing, we'll be surrounded by an exponentially growing number and variety of intelligent, interconnected devices. But the challenges associated with developing, maintaining and supporting increasingly sophisticated system architectures and protocols also will grow exponentially.

Looked at from this perspective, the emerging popularity of Java technology as a well-supported application and services framework for devices, offers an enticing possibility of ready-to-use software components that can be used along with embedded Linux, to speed and simplify device development and to enhance the capabilities of the end product.

Resources

Rick Lehrbaum (rick@linuxdevices.com) created the LinuxDevices.com "embedded Linux portal". Rick has worked in the field of embedded systems since 1979. He cofounded Ampro Computers, founded the PC/104 Consortium and was instrumental in creating and launching the Embedded Linux Consortium.

Archive Index Issue Table of Contents

Advanced search

The Perspective from My Garage

Doc Searls

Issue #94, February 2002

Even now, mainstream media is getting the story all wrong—the backlash against Linux in the business press.

Linux For Suits

The Perspective from My Garage

Even now, mainstream media is getting the story all wrong—Doc reflects on the backlash against Linux in the press.

by Doc Searls

On my browser is a CNET piece titled, “Got Linux? Many Companies Say No”, by Sergio G. Non. The subhead reads, “News Analysis: Linux penguins are braying louder, but companies don't plan to adopt many of them in the near future.” The opening text paints fact as speculation, then shoots it down: “...because the economy is still weak, many tech observers believe that Linux—and its price tag of 'free'--will attract more businesses looking to cut costs. At least that's the theory. Practice indicates something else.”

The “practice”, however, is anything but. It's a Goldman Sachs survey of 100 technology executives who were asked to name their highest and lowest spending priorities. Top priorities were migrating to new versions of Windows, security software and UNIX servers. At the bottom were mainframes, supply-chain management software and Linux servers. Again, these were spending priorities. No practice involved. Also no respect for the virtues of Linux as an instrument of savings.

We finally run into actual practice about halfway through the piece, when the author reports that Amazon.com has just “saved millions by switching to Linux from UNIX in many areas of its business”. Then we read quotations from an IT consultant who says, “Many of our clients consider Linux to be a very real option for cost savings”, and “There's a definite ROI.” The same consultant later adds, “I would agree that many companies are scaling back forward-looking projects.... However, the feedback we get from our clients is that they are very interested in Linux.”

In fact, the weight of evidence in the piece actually favors Linux, on the whole. But that's not the story this CNET writer wanted to tell. There's a backlash against Linux right now in the business press, and this guy was just doing his part. To be fair, there's a backlash against everything associated with the dot-com bubble, from venture capitalists and day traders to the countless wacky ideas that attracted hugely speculative investments.

But I think the backlash hits Linux worse for a number of reasons. One reason is that there aren't many pure Linux companies left. The Business Tux population has thinned to the point where it looks like an endangered species, even while a lot of big companies like IBM are crooning their love for Linux.

Last week I purged my collection of business cards collected over the last several years, and all but a few of them went in the trash. The ones I tossed were mostly from companies that are now dead, absorbed into other companies or busy repositioning themselves as something else.

And another problem was the brief but spectacular moment Linux had in the spotlight on Wall Street, a spotlight that fell on Tux at the very peak of the Street's speculative fever. In late 1999, no fleece was more golden than Linux, even though very few investors knew what Linux did, other than appear to threaten Microsoft. Between August and December of 1999, Red Hat, Cobalt, Andover and VA Linux all had huge IPO successes. VA's debut run-up was the biggest in Wall Street history.

Today, the only one of those four companies that still proudly flies the Linux flag is Red Hat—and I say hats off to them. Cobalt was acquired by Sun. Andover was acquired by VA, which recently filed to change its name to VA Software; they've even rid its stock of the once-valuable LNUX name.

The larger problem was that too many dot-com companies were not companies at all. They were projects—ventures. They were like those businesses with false fronts that lined the streets of mining towns built overnight in the Old West and abandoned almost as fast when the lode ran out, if it was ever there in the first place. From the street they looked like real businesses in real towns, but in fact, everything in sight was a gamble.

But Linux isn't a gamble, not as a technology. But it also isn't a business—and that brings us to our third problem, which is trying to understand and explain it to real businesses. The Goldman Sachs study cited by that CNET article typifies the problem. Linux in its pure state isn't for sale. You can't look at its popularity in terms of sales.

Worse, prognostications have ways of flat-out sucking anyway. Yesterday I went through boxes in the garage looking for a document I've been saving for a dozen years because it was so spectacularly wrong, right from the minute it was printed. I couldn't find it, even though it was thick as a phone book and chock-full of evidence for the continued growth of OS/2, Novell and other brand-name LANs, the likely success of AT&T in the computer business, the continued growth and proliferation of a dozen different UNIX brands, especially AT&T's original UNIX, recently bought by Novell and rebranded as UnixWare, a lot of stuff about Lotus Notes, and the continuing share losses for Apple. There was nothing, of course, about the Internet, and nothing about Linux. There was no idea that a PC would ever be hooked up to anything more significant than a printer and a fileserver.

But I did find some wonderful other nuggets. A *Sports Illustrated* from November 30, 1981, had a cover story on the preseason, number one North Carolina Tar Heels, with four of the team's stars: James Worthy, Matt Doherty, Jimmy Black and Sam Perkins. Not shown is the player that would emerge not only as the team's biggest star, but the best player in the history of the sport: Michael Jordan. And there was a *Popular Electronics* from July 1976. “Exclusive! Now You Can Build a High-Quality Intelligent Terminal!” it says. I page through it. There are a lot of ads for CB radios and stereo gear, electronics courses and antennas. But near the front is a two-page spread, the headline of which reads, “Imagine a Microcomputer”. Below is a box with LEDs and switches on the outside and an inside that's all backplane and power supply. The copy went on:

> Imagine a microcomputer with all the design savvy, ruggedness and sophistication of the best minicomputers.
>
> Imagine a microcomputer supported by dozens of interface, memory and processor option boards. One that can be interfaced to an indefinite number of peripheral devices....

It was an ad for the MITS Altair 8800-b. It didn't succeed, but the vision sounds mighty familiar, doesn't it?

> **Doc Searls** is senior editor of *Linux Journal*. His monthly column is Linux for Suits. He is also a coauthor of *The Cluetrain Manifesto*.

email: doc@searls.com

Archive Index Issue Table of Contents

Advanced search

Dealing with Patents in Software Licenses, Part II

Lawrence Rosen

Issue #94, February 2002

How to determine what you can live with regarding patent-retaliation clauses.

Last month I wrote about different categories of software patents and their effects on open-source and free-software licenses. I left the final category, licensee patents, for this follow-up article. This category is more subtle than the previous two. Here, we are dealing with a licensee's own patents or, perhaps even more important for derivative works, the patents of downstream sublicensees of the open-source software. Licensors sometimes include a patent-retaliation clause in their licenses to help defend themselves against infringement claims. In essence, a patent-retaliation clause says that if a licensee (or a downstream sublicensee) sues the software licensor for patent infringement, the license to the software terminates; a licensee or sublicensee can't both use the software and also sue the licensor for patent infringement.

One open-source software licensor justified its use of a patent-retaliation clause this way: “Maintaining the defensive use of patents will minimize unfairness.”

There are two general forms of a patent-retaliation clause. The first, a so-called weak patent-retaliation clause, says something like this: if you take a license for my software, and you later assert your patent against me relating to this software, then your license for my software is terminated. The same patent-retaliation clause usually applies to your sublicensees; if one of your sublicensees sues your licensor for patent infringement, your sublicensee's license to the software is terminated.

I personally support weak patent-retaliation clauses in licenses because I think they fairly balance the interests of the licensor and licensee. Licensees and their sublicensees should not be able to benefit from free and open-source software, while at the same time forcing the licensor to pay royalties for patents embodied in that very software.

A so-called strong patent-retaliation clause says something like this: if you take a license for my software, and you later assert your patent against me relating to anything at all, then your license to my software is terminated. Again, this patent-retaliation clause also usually applies to your sublicensees.

I personally believe that strong patent-retaliation clauses harm the Open Source community far more than they help licensors, because companies with large patent portfolios will resist adopting open-source software if the software licenses effectively allow licensors to infringe the licensees' (or sublicensees') other unrelated patents with impunity. Indeed, as a licensee of open-source software, you may find that a strong patent-retaliation clause inhibits your ability to disseminate derivative works. Your downstream sublicensees may refuse to accept such a virus that affects their ability to enforce their unrelated patents against your licensor.

Apple is an example of a company that insists upon including strong patent-retaliation clauses in its licenses. If a licensee adopts certain Apple software, the licensee's use of that Apple software is conditioned upon the licensee not suing Apple for patent infringement in any other matter. So to the extent that Apple's software becomes widely used, perhaps even indispensable, in a licensee's company, Apple can now infringe any of that licensee's other patents without fear of an infringement suit. A company with a large patent portfolio might be reluctant to adopt Apple software if it means, in effect, permitting Apple to use all of its other patents. In fact, I probably would warn any client of mine to consider avoiding Apple's open-source software for that very reason.

I used the terms weak and strong to describe the two forms of a patent-retaliation clause, but those terms may convey, in common usage, the wrong idea. I don't mean weak in the sense of lacking strength, energy or vigor. And I don't mean strong in the sense of either physical power or economic or financial robustness. Rather, what I intend to convey is the degree of pressure that can be exerted by the licensor to defend his or her interests against a licensee. A weak patent-retaliation clause only can be asserted in limited circumstances for a relatively small set of patent challenges. A strong patent-retaliation clause, on the other hand, can pressure a licensee to refrain from a patent challenge even for unrelated patents.

Whether you license your software to others or take licenses to software for your own use, you should make sure that you can live with any patent-retaliation clause you find in the license. If, as a licensee, you don't have (or intend to have) any patents that you can assert in retaliation, you may be able to live with either of the patent-retaliation clauses. On the other hand, if you intend to sublicense your derivative works, consider what form of patent-retaliation clause your sublicensees can accept. And as a licensor, consider how you later may be able to respond to a patent threat. Consider including a patent-retaliation clause if you need to use your software to help defend yourself against your licensee's patents.

Legal advice must be provided in the course of an attorney-client relationship specifically with reference to all the facts of a particular situation and the law of your jurisdiction. Even though an attorney wrote this article, the information in this article must not be relied upon as a substitute for obtaining specific legal advice from a licensed attorney.

email: lrosen@rosenlaw.com

Lawrence Rosen is an attorney in private practice in Redwood City, California (www.rosenlaw.com). He is also executive director and general counsel for Open Source Initiative, which manages and promotes the Open Source Definition (www.opensource.org).

Archive Index Issue Table of Contents

Advanced search

iXtreme 1350

Alan Zeichick

Issue #94, February 2002

A review of the iXtreme 1350.

Review

iXtreme 1350

Reviewed by Alan Zeichick

It's a whole new line of business, from a name (or a couple of names) that you'll surely remember. In 1996 the company was called Telenet Systems. It was renamed BSDi in 2000, and then in April 2001, it became iXsystems, Inc. What happened to the Berkeley Standard Distribution, you ask? Well, that's gone, sold to embedded software giant Wind River Systems, Inc. Today, the company is focused on building and selling custom low-profile, rackmounted servers, available with your choice of Linux, Windows or BSD (licensed back from Wind River). It's quite a change from the BSDi we used to know.

The new hardware product line consists of a range of Intel-based servers, ranging from an inexpensive 1U (that's one rack unit or 1.75" × 19") model with a single Celeron processor, to a 4U (7" high) system with four Intel processors. We recently reviewed a late prototype of their mid-range system, the iXtreme 1350, which is a 1U, dual-processor server optimized for maximum rack density in a web hosting facility or corporate data center, and given our initial skepticism about the OS-vendor-turned-box-builder, we were impressed with what we found.

The system we reviewed was equipped with dual 1GHz Pentium III processors, 1GB of RAM and three Seagate Cheetah 9.1GB Ultra3 SCSI hot-swappable hard drives. This is a common hardware configuration; all of the major server players, including Compaq, Dell, HP and IBM, also manufacture dual-processor 1U servers with essentially the same specifications; though Compaq ProLiant

DL360 and IBM xSeries 330 models only have two internal hard drives instead of three.

iXsystems equipped the server with the other hardware that you'd expect in a product in this class: built-in CD-ROM and floppy drives, serial and parallel ports, onboard video, two USB ports (fairly worthless for a server, unless you want to hang a USB printer off it) and dual 10/100 Ethernet connections. No surprises there.

We were surprised, however, that iXsystems chose to give the iXtreme 1350 only a single 64-bit PCI expansion slot. In the case of our review system, that slot was already filled with an Adaptec 2100S SCSI RAID controller. That's a decent board, which has a single Ultra3 SCSI channel—more than adequate for the three onboard hard drives. The Adaptec board also has an external SCSI connector, so you could hook it up to an external storage box to bring the number of hard drives connected to the server up to a maximum of 15.

Why is this a problem? In a rackmounted environment, many IT professionals may choose to connect their high-density servers up to a storage area network (SAN). Today, that requires the use of a Fibre Channel network adaptor. Without an open PCI slot that's not an option for the iXsystems 1350. Increasingly, many network managers are also connecting their servers via Gigabit Ethernet, rather than 10/100 Ethernet. Our lab also has a gigabit backbone, and we usually put Asanté FriendlyNet GigaNIX or Intel Pro/1000 network cards into new servers and hook them into a gigabit copper switch. In fact, when both Fibre Channel SAN access and Gigabit Ethernet networking are required, and there's only one open slot, we use InterPhase's SlotOptimizer 5570, which has both connections on a single PCI card.

Thus, with the iXtreme 1350, you can have SCSI, Fibre Channel or Gigabit Ethernet—choose only one. Competing servers either have two PCI slots, or have one PCI slot but place the SCSI controller on the motherboard. (Note that iXsystems will sell you either the RAID adaptor or gigabit adaptor, but doesn't offer a Fibre Channel adaptor.)

One other complaint about the hardware: the box's cover and PCI adaptor are held in place with tiny 3/8" long Philips-head screws. The other server manufacturers have learned that in a high-density, rackmounted environment, screws are bad news. Depending on the angle you're working on, you can have trouble getting them out, drop them when trying to put them back or even have them fall inside a server. All of the major manufacturers use catches, buttons or thumbscrews to hold down the cover, so you can slide it off or swing it open without using any tools. They also use catches or latches to hold the PCI cards in place, again making it easier to swap them out while the server is still inside the rack.

By using screws, iXsystems saved some dollars on their design and manufacturing but have made the server more difficult to service in the field. If you purchase this model, our advice is that you remove the four screws holding down the cover before installing the server inside the rack. Unfortunately, you can't remove the screw holding the PCI slot together without making the board wobble, so you'll just have to live with that.

The Software Side

iXsystems offers this model server with your choice of Windows 2000, FreeBSD 4.2 or Red Hat Linux 7.1 pre-installed. (The company's spec sheet says that the server also can come with BSD/OS and Solaris, but the on-line configurator didn't present them as options.) Guess which one we chose?

Red Hat booted up just fine on the server and connected instantly to our LAN. It had all the open-source applications and goodies included with Red Hat's 7.1 Professional Server distribution; yes, I know that 7.2 has been out for a while, but 7.1 is what iXsystems offers. The software appeared to be competently installed, and it took little time to bring up Apache and put a web site on-line. We only had the review server for two weeks, but during that time, it ran the web site fine, handling traffic (50 simulated users, sent over by Rational Software Corp.'s SiteLoad software on another) without a hiccup or even a sneeze. It's a Linux server. What more can you say?

A Good Value

Out of the box, the iXtreme 1350 represents a good value and has the secondary benefit of being from a company that truly understands the Linux/ UNIX universe. As equipped, the server carried a list price of $3,319 US, according to the company. That includes 90 days of e-mail/phone support for the operating system, and three years of what the company calls standard hardware support. That means, if the server breaks, you ship it to their depot, and they'll fix it and send it back. Or, if you can identify a broken part, they'll ship a replacement out to you.

The company really socks it to you, however, if you want more OS support—increasing the OS support to three years costs another $1,400. Their charges for on-site hardware support are more reasonable and are worth getting if the server is critical to your business: $450 US for three years of next-day or $770 US for three years of same-day support. So, a system with no extra OS support, but with next-day on-site support, would cost $3,769 US.

For comparison, we looked at two similar systems, each dual-processor 1U servers with Linux, RAID controller and three-year, next-day on-site support. The Dell PowerEdge 1550, with the same hardware except for three 18GB drives (Dell no longer sells 9GB drives), came through at $4,041 US. That's nearly a wash. The Compaq ProLiant DL360, with dual 9GB drives, was an astounding $7,461.

Based on hardware, software and pricing, we're impressed with the iXtreme 1350; the company has done a nice job, and we'd have no hesitation in deploying them or recommending them to clients, once they take the cover screws out. Perhaps iXsystems will have trouble living down their BSDi heritage and making the cultural transition from being an operating-system brand to an off-the-shelf server manufacturer, but in our opinion, they're off to a fine start.

Product Information/The Good/The Bad

Alan Zeichick is a technology analyst in the San Francisco Bay Area who focuses on networking and software development. Reach him at zeichick@camdenassociates.com.

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #94, February 2002

Readers sound off.

Backups with a Beat

Your December 2001 cover promised “Blazing Backups”, but you missed an opportunity for an artistic statement. It's not too late! Here's a suitable addendum to the article (to the tune of the “Blazing Saddles” theme):

> He made a blazing backup, He used a SCSI drive,So should the system crack up,His data will survive.He streamed with tar and compressed with zip,So fast the routers would shriek,He made his blazing backupAt least three times a week.

—JCool++The Dapper Rapper

Some Ultimate Advice

In your article “The Ultimate Linux Box 2001: How to Design Your Dream Machine” (unabridged web version, available at /article/5420), you wrote:

> The SB Live! seemed to work with the stock emu10k1.o sound module in Red Hat 7.1, but as it turns out it can't run the earphone-out jack on the LiveDrive.

Actually, it can—it just isn't set up to do so by default. I own one of these cards, a good pair of headphones, and a cheap pair of speakers, so I had reason to look into this. Here's how I got it working. First, download the drivers from opensource.creative.com:

```
cvs -d ':pserver:cvsguest@opensource.
creative.com: /usr/local/cvsroot'
login [use the password 'cvsguest']
cvs -d ':pserver:cvsguest@opensource.
creative.com: /usr/local/cvsroot'
co emu10k1
```

Everything there is under the GPL and changes here get folded into the kernel tree, so there's no real point using this driver over the kernel one. However, there are some utilities included that let you (among other things) enable different inputs/outputs. So compile and install their emu10k1.o if you want, but there's no need. What we're after is **make tools**. This gives you all sorts of tools for doing fancy things with the card, most of which I don't understand. The only one you need to get the headphones working is emu-dspmgr, located in the utils/mixer directory. With it, you can pipe your choice of inputs to your choice of outputs, e.g.:

```
   emu-dspmgr -a'Pcm L:Phones L'
   emu-dspmgr -a'Pcm R:Phones R'
   emu-dspmgr -a'CD-Spdif L:Phones L'
   emu-dspmgr -a'CD-Spdif R:Phones R'
```

--Andrew Bishop

File Creation with syslogd

I've been a longtime reader; while most issues haven't gotten the attention they've deserved, I do find *LJ* to be a useful source. In your December 2001 issue, Mick Bauer states that syslogd will create missing files. This certainly is not true of my version of syslogd, though I admit I haven't upgraded this box in quite some time. If the box isn't accessible, security patches aren't as important. The docs of the current version suggest that syslogd can create files, but my version also makes that claim, and I've verified it won't. (Oh, and if it did, it would be considered a security hole by some, myself included.)

—Ed Grimm

Bauer replies: As it happens, I tested and verified syslogd's file-creation behavior while working on the article; it works fine on both SuSE 7.1 and Red Hat 7.0. Personally, I don't consider this a significant security risk. At worst it's a denial-of-service exposure, but that's why it's smart to give /var its own partition, i.e., in case logs fill up the filesystem, whether by accident or attack.

Great Webmin Article

I have been using Webmin for a couple years now to administer my servers. It is a wonderful and powerful tool. I was very pleased to read about it in your December 2001 issue. Mr. Elmendorf did an great job. I hope to hear more on Webmin in the future in the pages of *LJ*--for example, how to add modules and such. Great work!

—Jody “JoLinux” Harvey

An Invalid Car?

While reading the informative column (Geek Law, *LJ*, November 2001), I was amused to note the example of LeCar as an invalid trademark for a car—you see I used to drive a Renault LeCar from 1978-1983. I don't know if the name was trademarked. I expect it wasn't strictly necessary in this case, as who else would name a car LeCar except Renault?

I suppose this demonstrates the difference between branding and trademarks. Whether trademarked or not, LeCar was a Renault brand. There was no point in naming another car LeCar. On the other hand, GTO was originally a Ferrari usage (I think) but was appropriated by Pontiac later. By the way, GTO is an acronym (Gran Turismo Omologato), which according to AltaVista means “great accredited tourism”, which I suspect doesn't express the flavor of the phrase.

In most countries the LeCar was known as the Renault 5; LeCar was only used in the US as far as I know. A fun little car—I regularly beat VW Rabbits off the line—a slow-motion drag race.

It was the first new car I ever bought, just days after getting hired at Tektronix, for my first career job. It survived over 100,000 miles with only two major fixes (a blown head gasket and a bad wheel bearing), which was something of an accomplishment in those days.

—Gary Bickford

Archive Index Issue Table of Contents

Advanced search

Advanced search

UpFront

Various

Issue #94, February 2002

Stop the Presses, *LJ* Index and more.

UpFront

BFNB

Early last year Don Marti forwarded an exceptionally clueless e-mail from a public relations person. His subject line said, “Bad flack! No bisquit!” I nearly died laughing, since (I hate to confess) I've done serious time as a flack myself. Since then BFNB has entered the private lexicon here at *Linux Journal*. And, now we'll share a few choice nuggets with the rest of you:

- “If you would like to speak to __ executives about __'s future plans and how Linux 7.1 affects the Internet, network environments and the IT world...”
- “__ also doesn't discriminate against computers using the Linux operating system. The __ has an open architecture, which means Linux users can customize it for their networks by loading their own proprietary software on top of __'s software.”
- “I?am taking?the?liberty?of?reintroducing? you?to?__in?case?you?did?not?receive?our?previous?correspendence.?This?is ?an?excellent?opportunity?for?the?serious?investor?who,?like?us,feels? the?energy?sector?is?the?place?to?be?in?these?times?of?rising?oil?and ?gas?prices.”
- “The 70s were cool. Earth, Wind and Fire, tank tops and mood rings were all the rage. Thirty years later, in 2001, the 70s remain cool. So don't throw away your old low-rider jeans, choker necklaces or mainframe computers—what's old is now cool, and it's called retro. __ lets you retro-fit your old technology into today's hippest platform.”

That last one never said a word about what __ was, or what it did. We must assume, however, that Linux was indeed the hippest platform.

—Doc Searls

Linux-Based Googlestructure

I remember when some Linux geek told me about Google several years back. He said the new search engine, then in public beta, was going to kick butt because they were building it on Linux servers. I didn't believe him. At the time my preferred search engine was HotBot, which consistently outperformed all the other search engines at what I cared about most: finding documents based on text strings, some of them buried deep in a page's text. HotBot recently had supplanted AltaVista as my first-choice search engine. Before AltaVista I liked InfoSeek (I was one of those few who actually subscribed to InfoSeek's services). And before that I liked Lycos, which was still an academic project at Carnegie-Mellon. Eventually HotBot lost out to FAST, the BSD-based engine with an utterly mismatched URL: alltheweb.com. But resistance was futile. Google got me.

At first I didn't like Google because it was too simple and too insistent about knowing what I wanted. I hated that. Still do. But I came to love Google, because dammit, they did seem to know what I wanted—not always, but often enough. Now, like most of us, I hardly use anything else.

Today the other engines are also-rans. With each new step forward in functionality (image and newsgroup searches, file-type searches, additional languages), Google seems to leave the others farther and farther behind.

I hadn't spoken to the Google folks in a while, so thought I'd check in and get some specifics, including the answer to the most existential question of all: are they making money yet? So I went to my old neighbor Cindy McCaffrey, Google's vice president of marketing, who told me:

> We're profitable. Advertising has been a big contributor to that profitability. Both of our ad programs (Premium Sponsorships, AdWords) are ramping up quickly. We have thousands of advertisers and have just begun expanding our advertising internationally with the opening of small ad sales offices in the UK, Japan and Germany.

This was particularly interesting to me because the ads are a lot like newspaper classifieds, which are the only form of advertising for which there is high reader demand. Like classifieds, ads on Google are unobtrusive and contain no graphics. When I asked one advertiser how well the ads work, he said, "Very well. All our advertising is on Google." In fact, they work so well that he advertises in spite of his objection to Google's policy of seeking patents for its technologies, a practice he despises. Cindy added:

> The keyword-targeted approach is working well for us. Our click-through rates average about 2+ percent, about four to five times higher than the industry average for traditional banner ads. We also offer search services to other companies such as Yahoo!, Cisco, Sony, etc.—about 130 customers in about 30 countries. The split between these two revenue sources is roughly 50/50.

I asked if the company's mission had changed at all. My guess was that it hadn't, since it never succumbed to the distractions that trivialized vanquished competitors: stock prices, sports scores, cross-promotions with entertainment sites, etc. She said no, their mission is what it's always been: "To organize the world's information, making it universally accessible and useful."

It might not be a stretch to say that Google has, for many of us, become part of the web's infrastructure—its search interface. To get some sense of how far that interface reaches, I asked Cindy to send me some numbers. Here they are:

- data centers: 4
- Linux computers: >10,000
- searches per day: >150 million
- index of web pages: >1.6 billion
- image base: >330 million
- Usenet messages: >650 million
- newsgroups: >5,000
- language subsets in the index: 28
- international domain sites: 23
- PDFs: >22 million

Many of those are "most on the Web", she modestly added. But she declined to confirm the hypothesis offered by that geek who turned me on to Google in the first place: that Linux was the reason. Guess we have to draw our own conclusions.

—Doc Searls

LJ Index—February 2001

1. Position of “white box” units among top-selling PC “brands”: 1
2. Market share range of “white box” PCs: 50-70
3. Millions of player pianos sold in the US alone by 1930: 2.5
4. 1930 US population in millions: 123.2
5. Linux training revenue in millions of dollars in 1999: 10.3
6. Linux training revenue in millions of dollars in 2001: 56
7. Projected Linux training revenue in millions of dollars in 2004: 285
8. Size in billions of dollars of the system-level training market by 2004: 2
9. Low end of projected Linux percentage share of the system-level training market by 2004: 5.8
10. High end of projected Linux percentage share of the same market: 15.3
11. Projected compound annual growth rate of Linux support services revenue from 1999-2004: 86.9%
12. Amazon.com technology expenses in millions of dollars prior to migration to Linux: 71
13. Amazon.com technology expenses in millions of dollars after migration to Linux: 54
14. Percentage cut in Amazon technology expenses in migration to Linux: 25
15. Cost relative to UNIX of 1,000 users tapping into a Linux server: 1/3 to 1/2
16. Value in millions of dollars to IBM of tools donated to public domain by IBM via the Eclipse.org Project: 40
17. Number of software tool suppliers working on Eclipse software at the formation of Eclipse.org on November 6, 2001: 150
18. Number of individual developers involved with Eclipse at the same point: 1,200
19. Number of hosted projects on SourceForge.net on November 11, 2001: 29,253
20. Registered users of SourceForge.net at the same time: 290,500

Sources

1-3: Jon “maddog” Hall of Linux International

4: US Census Bureau

5-11: International Data Corp. (IDC)

12-14: CNET

15: Dan Kusnetzky of IDC, in CNET story

16-18: IBM

19-20: SourceForge.net

Stop the Presses: DMCA 2, Free Speech 0, but the FTC Offers Hope

In the last few days (as we go to press in mid-December 2001), the cause of free speech, which has been dear to the heart of the Linux community—and without which we might not have either Linux or the Internet—has taken a couple of heavy hits in the courts. The only ray of hope comes from the FTC.

In both court cases the DMCA (Digital Millennium Copyright Act) prevailed. The DMCA, which was signed into law in October 1998, expands the scope of copyright to criminalize circumvention of copyright schemes, among other things.

In Felten vs. RIAA, Professor Edward Felten (currently on leave from Stanford) and others sued the Recording Industry Association of America, claiming that the RIAA used threats of lawsuits to prevent him from presenting his work at an academic conference. Felten had planned to show how he and his researchers had disabled digital watermarks but backed down after a threatening letter from an RIAA lawyer. RIAA officials later said they had not intended to sue. The case was dismissed by US District Court Judge Garrett E. Brown, who found that Felten had no legal complaint.

Context: Dmitry Sklyarov, a Russian academic was imprisoned and is still awaiting trial for DMCA violations he allegedly committed when he gave a presentation at a conference where he detailed weaknesses in Adobe's eBook technology software. At the behest of Adobe, Sklyarov was arrested in his hotel in Las Vegas on July 16, 2001 while preparing to return to Russia. He spent three weeks in jail and still awaits trial, even though Adobe has withdrawn its support for the case.

In a statement, Cary Sherman, senior executive vice president with the RIAA, said:

> We are happy that the court recognized what we have been saying all along: there is no dispute here. As we have said time and again, Professor Felten is free to publish his findings.

In a press release following the dismissal, Electronic Frontier Foundation (EFF) Legal Director Cindy Cohn said:

> Since the government and industry cannot agree on what the DMCA means, it is not surprising that scientists and researchers are confused and decide not to publish research for fear of prosecution under the DMCA....Regardless of specific government or industry threats in the past, scientists should not have to experience the ongoing chilling effects of this vague digital copyright law.

In Universal vs. Reimerdes, the 2nd US Court of Appeals affirmed a district court ruling against the defendants. The plaintiffs included Universal Studios, plus Tri-Star, Disney, 20th Century Fox, Paramount, Columbia Pictures and MGM. The defendants were Eric Corley and *2600* magazine, which had published DeCSS, a DVD decryption program that has circulated widely on the Net (*2600* provided a download) and that allows users of Linux computers to watch DVDs. Corley's attorneys argued that DeCSS was protected as free speech. The studios argued that harm to their industry outweighed free speech protections. The court agreed with the plaintiffs. The EFF's Cindy Cohn, who helped represent Corley in the case, said, “I think it's a setback for free speech. It appears that the court is upholding censorship of the magazine on-line.”

It is also significant that the DMCA, in the words of Eric S. Raymond of the Open Source Initiative, serves to “protect the cartelization” of playback devices. Jon Johansen and MoRE (Masters of Reverse Engineering) in Norway developed DeCSS basically so DVDs could be played back on Linux devices—something the movie studios and their partners in the consumer electronics business hadn't bothered to deliver.

On one positive note, the US Federal Trade Commission announced hearings on “Competition and Intellectual Property Law and Policy in the Knowledge-Based Economy”, starting in January 2002 (when this issue of *Linux Journal* should be hitting the streets). You can learn more about it and submit written comments by following directions from the Federal Register at this URL: www.ftc.gov/os/2001/11/ciphearingsfrn.htm.

—Doc Searls

They Said It

In translating *Der Spiegel* into English via Babelfish, I discovered that “anthrax” translates to “spleen fire”.

—Anita French

Really, the reason you see open source there at all is because we came in and said there should be a platform that's identical with millions and millions of machines.

—Bill Gates

Linux is the long-term threat against our core business. Never forget that!

—Brian Valentine, VP Micosoft Windows Division

I've imagined doing software backwards—and it almost works. Backwards 1.0 has a ton of great features. With each release it has fewer features, until, one day, it's down to its core, the bare few features that make it a killer app.

—Brent Simmons

We've recently...found that Linux—if you look at the overall cost of ownership including the hardware, software, staffing, and purchasing and retirement costs —ends up being significantly less expensive than UNIX over a three-year period for things like web serving.

—Dan Kusnetzky, International Data Corp. (IDC)

A teacher who establishes rapport with the taught becomes one with them, learns more from them than he teaches them. He who learns nothing from his disciples is, in my opinion, worthless. Whenever I talk with someone I learn from him. I take from him more than I give him. In this way, a true teacher regards himself as a student of his students. If you will teach your pupils with this attitude, you will benefit much from them.

—M. K. Ghandi

He had the startling individuality of a man who had never owned a television.

—Gerald Hannon, in an obituary for Michael Stanley Kibbee, founder of Cemetary.org.

The soul of democracy has been dying, drowning in a rising tide of big money contributed by a narrow, unrepresentative elite that has betrayed the faith of citizens in self-government.

—Bill Moyers

Our analysis is that many users of Linux don't want their organizations or their competitors knowing what they're running. They'd rather get kudos for a job well done than criticism for how they got the job done.

—Dan Kusnetzky, International Data Corp. (IDC)

Archive Index Issue Table of Contents

Advanced search

This Ain't Your Dad's Office

Richard Vernon

Issue #94, February 2002

Working from home—it's not just the cool thing to do—it's the right thing to do.

From the Editor

This Ain't Your Dad's Office

Working from home—it's not just the cool thing to do—it's the right thing to do.

by Richard Vernon

The idea of the small and/or home office is one that is receiving an increasing amount of attention. As technology and the multiple demands on our time make working from home a better option, the home office is something of which we are seeing more. And, as many once medium-sized businesses have shrunk with the current economy, the small office is becoming a more frequent element in the business landscape.

A principal attraction of the home office is avoiding a long commute. My own commute involves riding a bicycle seven and a half miles in the Seattle rain (though I must confess that it's only slightly uphill in one, not both, directions) and an hour ride on a ferryboat each way. I always thought in my narrow-minded way that the advantages of working from home were limited to the personal luxury of working in my socks and underwear and saving the commute time. But of course I was missing the big picture. Last month we ran an interview that our publisher, Phil Hughes, did with Costa Rica's Minister of Science and Technology Guy F. de Téramond. Téramond is a key figure in bringing universal internet connectivity to Costa Rica. He mentions that part of his motivation for this project was to allow more Costa Ricans the opportunity to telecommute, not only for improved quality of life, but because the increasing population in urban centers was placing too great a stress on the transportation infrastructure. An increase in telecommuters should lighten that load. Living in the Seattle area, which has one of the worst traffic problems in the world, I now work from home when I can, not for selfish reasons, but because hey—I'm just doin' my part to make the world a better place.

Our two feature articles this month should be appealing to anyone who uses Linux to work from home. If Don Marti's prediction that used laptop computers will be down to eight dollars by the time you read this, then maybe I can set up the wireless home network he describes in his article. Being as I share my current “home office” with the furnace and lovely asbestos wall hangings because any other room is too inconvenient a distance from my wife's computer for our collection of cat 5 cables, this would be a real benefit.

Rory Krause's article on printing documents located on a remote office system at home using ssh's port-forwarding feature eliminates yet another reason for not working from home. With the script included in Rory's article, you can print remote documents without having to use scp or print to a file.

Between the instructional content in both articles, you should be able make some substantial improvements to your small office and/or reap the benefits of working in your socks and underwear while saving the environment.

Richard Vernon is editor in chief of *Linux Journal*.

Best of Technical Support

Various

Issue #94, February 2002

Our experts answer your technical questions.

Restoring /dev

Since becomming a subscriber in 1998, I have been using the tips in the Best of Technical Support column. Sometimes I laugh at basic problems, thinking “this will never happen to me”--until now.

I am using Red Hat 5.0. A bad shell script accidentally removed my /dev! I shut down my Linux system and rebooted before restoring /dev from backup. So, I cannot restore the /dev from tape using my rescue diskette, because /dev/st0 is gone. What are the steps to recreate a basic /dev directory using my rescue diskette? After that, how do I restore the /dev with tar?

—Rene, rene.diependaele@ibbbs.be

Your rescue media also should have the default minimal set of devices, and you could copy them to your now-empty /dev directory with **cp -av /dev/* /mnt/dev/** (if your disk is mounted under /mnt). Then you can boot your system with **linux init=/bin/bash** at the LILO prompt, and you will need very few devices.

—Marc Merlin, marc_bts@valinux.com

Then, to restore st0, do a **mknod /mnt/dev/st0 c 9 0**.

—Christopher Wingert, cwingert@qualcomm.com

Extra RAM Makes System Flaky

I have a Soyo K7V Dragon with a 1,400MHz Athlon with BIOS settings for “optimized defaults”. I installed 1.5GB Nanya PC2100 memory and Red Hat 7.2.

The system boots and runs but crashes sporadically. No messages in syslog, just “restart....” When I remove 512MB of the 1.5GB, the system runs fine.

I've tried dozens of combinations of distributions, kernels and BIOS settings, as well as several disk configurations (hardware RAID, software RAID, no RAID, disk on Promise controller only, disks on Via controller only) to no avail. The distribution kernels do not recognize the via8233 southbridge. I made appropriate modification to via82cxxx.c and recompiled the kernels. The southbridge is found, but the system still crashes.

I finally replaced the memory, but the problem persists. The 512MB DDR sits burning a hole in my desk. It was expensive ($199/stick). Of course, it's cheaper now, but I'd still like to get this system running to its potential.

—Jim Peterman, jptr@msn.com

Did you replace all the sticks or just one? If only one, you might want to replace them all.

—Christopher Wingert, cwingert@qualcomm.com

It is possible that you have a defective memory slot on your motherboard, or that it's not reliably able to power more than two slots of RAM.

—Marc Merlin, marc_bts@valinux.com

PCMCIA Card Doesn't Work with 2.4

I've been receiving *Linux Journal* for a couple of years now and follow most of the technical things in there. But now I have a technical question I can't solve concerning the Teles PCMCIA/S0 card and kernel 2.4. I searched the Internet, newsgroups, etc., for solutions but found none. What I did find in the newsgroups are other people that have the same problem. Maybe you can help us out here.

When I used kernel 2.2.x I always used the Teles PCMCIA/S0 card. No problem at all. When compiling a new kernel, I also needed the extra PCMCIA card services and a patch downloaded from home.wtal.de/petig/ISDN, and everything worked fine. However, from kernel 2.4 this patch cannot be used anymore—it just won't compile. In the documentation of the kernel sources for 2.4, it says that they support the Teles PCMCIA/S0 card. But, whatever I try I can't get it to work. It always complains about not having the right I/O address, even when I use the IRQ and I/O that I used with the 2.2.x kernel release. The second thing is when inserting the card, I always get the message that the kernel looks for module teles_cs.o. This was the one from the home.wtal.de/petig/ISDN site.

Is there any way you can help me and all the people that want to use this ISDN card? I'm now using the standard Red Hat 7.1 version with all the patches applied.

—Andre Seesink, a.seesink@chello.nl

You might want to verify that the IRQ and I/O address you specify are still valid. Sometimes when you boot other OSes, I/O ports/IRQs on some Plug-and-Play devices are moved automagically.

—Christopher Wingert, cwingert@qualcomm.com

I haven't had much luck with the kernel PCMCIA support myself. David Hinds (the pcmcia-cs author) recommended that I try disabling PCMCIA in the kernel (you'll have to recompile your kernel with PCMCIA disabled), and that I try using the standalone pcmcia-cs package. My problems have gone away since I've done that. Yours may too.

—Marc Merlin, marc_bts@valinux.com

Unsupported Filesystem

I am using Red Hat 7.1 with Windows 2000 on my system. The primary partition is FAT16 (hdb1), and the secondary partition is NTFS (hdb2). Linux is installed on a second hard drive, which is connected as the secondary slave (hdd). While trying to mount the NTFS partition, I keep getting the following error:

```
The kernel does not support the ntfs fs.
```

The version of the kernel is 2.4.2-2. All my data resides on the NTFS partition. I would like to use Linux as the primary OS, without changing the partition structure.

—Nigel Pereira, pnigel1@hotmail.com

In order to mount and access this partition, you'll have to compile the ntfs support in your kernel, since it is not available by default. The process of configuring and compiling the kernel is documented in the Kernel HOWTO that usually can be found in your distribution or at the linuxdoc site (www.linuxdoc.org/HOWTO/Kernel-HOWTO.html). Be careful though to keep the last working version so you can boot back if something goes wrong. Your best solution would be to migrate that partition to FAT32, but if you require

NTFS, then you will need to compile yourself a new kernel. Enable experimental code to see the option for NTFS. Whatever you do, do not enable write support. It is guaranteed to hose your partition.

—Ben Ford, ben@kalifornia.com

I Have No FontTastic, and I Must Use WordPerfect

I am running Red Hat 7.0; I installed WordPerfect Office 2000, and I get the following message every time I attempt to open/run any WordPerfect Office 2000 products:

```
Unable to add FontTastic font server to the font
path. The font server is probably not installed or
not running. Correct the problem and try again.
```

I've gone to the Corel web site, sent them messages and still haven't resolved the problem. Is there any reader or staff member at *Linux Journal* able to help me?

—James H. Birdsong, jbirdsong@orbitworld.net

For some reason the WordPerfect team at Corel decided they needed their own font server. Unfortunately, this has some issues with XFree86 4. This is a known issue; search the Corel newsgroups for the solution.

—Ben Ford, ben@kalifornia.com

Two Questions, One Answer

How can I move my MP3 files from FAT32 to Linux's ext2?

—Cougar, cougaram@home.com

When I was using Mandrake, it would recognize and mount my windows drives that I have on a separate hard drive just fine. Now that I am using Red Hat 7.2, my windows drives are not listed, and I cannot mount them. Is there anything I can do to mount /dev/hda? Right now it is installed on dev/hdb.

—Glen Kingston, gkingstun@yahoo.com

You have to mount your Windows partition in Linux. To figure out what your Windows partition is use **fdisk -l /dev/hda**. Look for your Windows partition (FAT32 or NTFS), then mount your partition. For example, if your Windows partition is /dev/hda1, do the following as root:

```
   mkdir /dos
   mount /dev/hda1 /dos
```

Your Windows partition will appear under the /dos/ directory.

—Christopher Wingert, cwingert@qualcomm.com

New Products

Heather Mead

Issue #94, February 2002

Ximian Evolution 1.0, Matisse 5.0, XAO 1.0 and more.

New Products

Ximian Evolution 1.0

Ximian Evolution 1.0 is a personal and workgroup information management suite that integrates e-mail, calendar, contact and task lists into one application. Designed for mixed corporate computing environments, Evolution includes broad support for data exchange and communications standards that allow Linux and UNIX-based systems to be integrated directly into corporate networks and messaging systems. It provides support for SMTP, POP, IMAP and other messaging protocols, as well as peer-to-peer calendaring functions that can be shared with MS Outlook, Lotus Notes and other iCalendar-supported applications. The Ximian Connector for Microsoft Exchange 2000 add-on is also available.

Contact Ximian, Inc., 401 Park Drive, 3 West, Boston, Massachusetts 02215, 617-375-3800, www.ximian.org.

Matisse 5.0

Fresher Information Corporation announced the release of Matisse 5.0, database software for rapid development and deployment of object applications and web services. Matisse combines native object support with server-based SQL, thereby eliminating object-relational mapping. ODBC and JDBC support is provided for accessing external data sources and application development and reporting tools. Matisse 5.0 also supports Solaris, NT/2000 and FreeBSD platforms; SQL, UDDI and XML standards; and a range of language bindings, including Java, C, C++, Python, Perl and PHP. A free developer download is available from the web site.

Contact Fresher Information Corporation, 575 Market Street, 13th Floor, San Francisco, California 94105, 415-356-8100, www.fresher.com.

XAO 1.0

XAO 1.0, web-services software based on Apache that enables standardized data integration from sources such as relational databases, legacy applications and other application servers, is now available from XAO, Inc. The XAO 1.0 Foundation Server provides an API on top of any relational database, providing an object-level view while retaining the speed of relational queries. The Foundation Server also performs storage and deep-search functions for other XAO modules and web applications, including e-commerce. XAO 1.0 ships with a set of unit tests that allow for application customization without worrying about leaving established code.

Contact XAO, Inc., 221 East Walnut Street, Suite 102, Pasadena, California 91101, 877-796-7437 x24, www.xao.com.

Accelerated X Summit 2.0

The Accelerated X Summit 2.0 series of downloadable graphics drivers are now available. The new drivers are based on an OpenGL 1.2.1-compliant rendering pipeline, and include 2-D and 3-D support for more than 30 cards and laptops. Series 2.0 is available in four series: Desktop, Laptop, Multihead and Workstation, with four editions under each. New and enhanced features in v2.0 include: 3-D stereo on most cards; Color Magic, a graphical utility for calibrating system color; Video Window, to allow viewing MPEG or other video sources; XiG Direct Access, for direct access to the graphics hardware by OpenGL applications; and DualView, a two-screen display available on most laptops and cards.

Contact Xi Graphics, Inc., 1801 Broadway, Suite 1710, Denver, Colorado 80202, 800-946-7433, www.xig.com.

AdminForce CGI Auto Audit

LinuxForce, Inc. announced AdminForce CGI Auto Audit, a CGI script analyzer that manages script structure and identifies potential security deficiencies. The automated routines can analyze hundreds of scripts per day with the accuracy of a traditional line-by-line audit. In addition, a library of metacharacter “scrubbing” routines are provided to assist with script cleanup operations, and adjustments can be made for false positives. Testing can be conducted remotely with CGI Auto Audit.

Contact LinuxForce, Inc., 100 Glendale Road, Upper Darby, Pennsylvania 19082, operations@linuxforce.net, www.linuxforce.net.

Covalent Enterprise Ready Server

Covalent Technologies' Enterprise Ready Server (ERS) is a web server product designed to minimize the resources required to deploy and manage Apache web servers. ERS offers a distributed graphical management tool that combines Apache 2.0 with a Java application server while improving security and reliability. It also offers a one-to-many management portal for configuring, managing and monitoring hundreds of Apache servers and thousands of virtual hosts, and support for HTTP, HTTPS and FTP. ESR comes with the complete binary for Apache 2.0; Tomcat 4.0 Java server; and Covalent's management portal, user tracking, logging, authorization and authentication, SSL, SNMP, PHP and mod_perl services.

Contact Covalent Technologies, 645 Howard Street, San Francisco, California 94105, sales@covalent.net, www.covalent.com.

xLswitch

SWsoft's xLswitch is a scalable and highly available traffic, bandwidth and content management solution for web, mail and internet applications. Aimed at service providers, e-commerce businesses and corporate enterprises, xLswitch allows them to provide uninterrupted, rapid and personalized secure service delivery to customers. xLswitch combines load balancing and monitoring with traffic management and bandwidth control. xLswitch works by moving most of the code into the OS kernel and using algorithms for TCP-splicing and other operations. Graphical, multiplatform management tools are included, as are an SDK and documentation for developers to utilize xLswtich's extendibility.

Contact SWsoft, 395 Oyster Point Boulevard, Suite 213, San Francisco, California 94080, info@sw-soft.com, www.sw-soft.com.