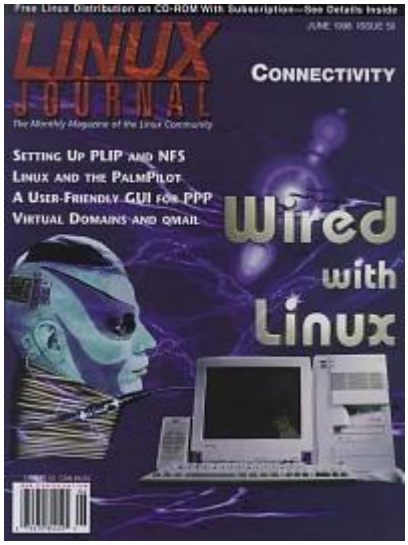
Linux Journal Issue #50/June 1998



Features

News & Articles

Reviews

WWWsmith

Columns

Archive Index

Advanced search

LINUX JOURNAL

Virtual Domains and qmail

Mike Thomas

Issue #50, June 1998

Here's a way to get control of your mail with secure, high performance and freely available software called qmail.

The program **qmail** is a secure and reliable replacement for **sendmail**; it was written by Dan Bernstein at the University of Illinois at Chicago. I was attracted to it for several reasons, the most important being that it runs under Linux.

qmail is substantially more secure than sendmail. The system is partitioned into several modules, minimizing the amount of code which runs as root. The /var/spool/mail directory is gone; incoming mail for a user is stored in the user's home directory, eliminating a nagging security hole. qmail gives you control over which mail you accept. You can selectively allow other hosts to use your system as a relay, blocking out all others.

qmail supports mailing lists with automated subscriptions, and these lists can be configured and maintained entirely by the user. No intervention is required on the part of the system administrator to create a new list.

qmail's performance is stellar. Dan Bernstein cites Red Hat Software as an example. Red Hat was running sendmail 8.7 on a 48MB Pentium and found their daily load of 70,000 messages was beginning to overwhelm the system. They switched to qmail on a 16MB 486/66, and their mail hub is now running fine, even on the less powerful hardware.

The reason I began looking into qmail as an alternative to sendmail is the fact that qmail supported e-mail for virtual domains correctly long before sendmail did. Those of you running several virtual domains on a single Linux host can rejoice. With qmail, the e-mail names you select for your virtual domains come from per-virtual-domain name spaces, rather than a single host-wide name space. This means you can have e-mail names like webmaster@domain1.com and webmaster@domain2.com simultaneously.

Drawbacks

The only problem we have had with qmail is the fact that its outgoing queue uses inode numbers in its database; this means the queue cannot be backed up on one machine and restored to another. When we have a disk failure, we must recreate an empty qmail queue directory rather than restoring from backup.

The fact that qmail is not sendmail implies some complications when installing add-on e-mail packages like majordomo. In general there are patched versions of these packages available for qmail.

qmail Installation and Configuration

The qmail sources are available at ftp://koobera.math.uic.edu/www/qmail.html and a lot of useful information is available at http://www.qmail.org/. Compilation and installation of qmail is straightforward. Those who balked at the sendmail.cf file will be pleasantly surprised at qmail's configuration. Everything is human readable and easy to understand. Some claim that sendmail.cf is human readable, but I would argue that point.

Once you have qmail configured and operational, you can start adding virtual domains. The rest of this article deals with virtual domains under qmail. All file and path names assume the default qmail installation.

Supporting E-mail for a New Virtual Domain

Set up the new virtual domain normally. Many of you will have already done this to support the virtual domain with other services like Apache **httpd**. Make sure there is an MX record in DNS to point mail for the virtual domain to the host running qmail.

Create a master user ID and home directory for the new domain. The *master user* is just a user who will control all mail for your virtual domain. I generally create a user ID for each virtual domain which the administrators of that domain can use to upload the content for their web site. qmail can use the same user ID.

Add a line to /var/qmail/control/virtualdomains for the new domain, directing mail for that domain to the user created above. If the domain is abc.com and the user is abc, an appropriate line would be:

```
abc.com:abc
```

Add abc.com to /var/qmail/rcpthosts to tell qmail you're willing to accept mail addressed to abc.com. Ensure abc.com does *not* appear in /var/qmail/control/locals/.

Once mail is directed to a user, it is controlled through a series of .qmail-*xxx* files in that user's home directory. Create the file ~abc/.qmail-default, to indicate user abc is willing to accept all mail directed to the abc.com domain.

Restart qmail and e-mail for all users at abc.com, i.e., john.smith@abc.com, webmaster@abc.com, etc. will now be received by the local user abc. I suspect this is not precisely what you had in mind, so read on.

Forwarding a Virtual Domain User's Mail

If mail for a new user in a virtual domain is to be forwarded to an existing on-site user or to an off-site user, you don't need to create an account for the new user. You can create a .qmail-*xxx* file in the virtual domain master user's home directory to forward the mail. The master user is the user we created above, who is currently receiving all mail for the virtual domain. For the address john.smith@abc.com, you create a file ~abc/.qmail-john:smith, containing the address to which John's mail is to be forwarded in this way:

```
&smith.john@home.boston.ma.us
```

Note that any periods in the user's Internet name are replaced with colons in the .qmail-*xxx* file name. The forwarding address which is stored within the .qmail-*xxx* file does *not* have periods replaced with colons.

POP3 Mail for a Virtual Domain User

If a user of a virtual domain will be picking up his mail using POP3, you must create an account and an incoming mail directory for him. The POP3 daemon, which comes with qmail, cannot pick up mail from an ordinary mbox formatted file.

```
# adduser jsmith
# chmod g-w ~jsmith
# chmod o-w ~jsmith
# cd ~jsmith
# maildirmake Maildir
# chown -R jsmith.users Maildir
```

The **chmod** commands in the above script ensure that no one can write to jsmith's home directory except jsmith himself. qmail enforces this requirement as a security measure, but it can be relaxed with a compile-time option—see **ALIAS_PATERNALISM** in the conf-unusual.h file.

Note that under Linux distributions which include the adduser command, like Slackware, you can do a **maildirmake** in /etc/skel, so new users will automatically get a Maildir.

As in the previous section, you need to create a .qmail-*xxx* file in the home directory of the virtual domain's master user to forward mail to each individual user. To forward mail for john.smith@abc.com to the local user jsmith we would create a file, ~abc/.qmail-john:smith, containing the line:

```
&jsmith
```

To indicate where his incoming mail should be stored, we would create a .qmail file in the home directory for jsmith, containing:

```
/home/jsmith/Maildir/
```

This step is required because the qmail POP server expects to find a user's mail in a specially constructed directory (the default name of which is Maildir), and we have to tell qmail to put it there.

Once you start storing incoming mail in a nonstandard place, you have to tell the local mail programs where to find it. The standard Linux mail programs cannot read mail from the Maildir format, so qmail includes several wrapper programs to move any incoming mail into mbox format (qail, qine, qlm, for mail, pine and elm respectively). You can rename the real mail user agents and link these wrappers to the usual names, so your users won't even see a difference. These wrappers need a bit of information to operate correctly. To take care of this, add this type of lines to the /etc/profile file:

```
export MAILDIR=$HOME/Maildir
export MAIL=$HOME/Mailbox
export MAILTMP=$HOME/Mailbox.tmp
```

The final thing you have to do is install qmail's POP3 daemon. It is split into three programs, one of which deals with user names and passwords. Those of you with shadow passwords installed will appreciate this modularity. A password checking program, **checkpassword**, which works with ordinary Linux /etc/passwd files, is available at the same URL as the qmail distribution. The POP3 line in your /etc/inetd.conf will have to be modified. How to do this is described in detail in the FAQ that comes with qmail.

If you feel the above changes are too disruptive, an alternative is to patch your existing POP3 daemon to look for a user's incoming mail in an mbox-formatted file in the user's home directory, rather than a similar file in /var/spool/mail. One such package is available at ftp://summersoft.fay.ar.us/pub/qmail/. The

only thing you lose by using a patched POP server rather than the POP server distributed with qmail is the much more reliable Maildir mail storage format.

Forwarding Virtual Domain User Mail Without a Master User

If you want to forward all mail for a new virtual domain, but you have no reason to create a master user ID for that domain (e.g., you're not providing web services), you can do this using the special *alias user* ID. Instead of adding the line **abc.com:abc** to /var/qmail/control/virtualdomains, add the line:

```
abc.com:alias-abc
```

This designates the alias user as the responsible party for all mail to the abc.com domain. qmail's default installation sets the alias user's home directory to /var/qmail/alias, so control of all e-mail for abc.com is done in this directory.

You can create a file ~alias/.qmail-abc-default to forward all mail for abc.com to a specific user. You can also create a series of files, like ~alias/.qmail-abc-webmaster and ~alias/.qmail-abc-john:smith, to forward mail for specific people at abc.com.

Note that the alias user (or any other user) can control mail for multiple virtual domains. To control abc.com and anotherdomain.org, put the following lines in the /var/qmail/control/virtualdomains file:

```
abc.com:alias-abc
anotherdomain.org:alias-anotherdomain
```

You'll need these files in the ~alias directory:

```
~alias/.qmail-abc-john:smith
~alias/.qmail-abc-nancy:jones
~alias/.qmail-abc-webmaster
~alias/.qmail-anotherdomain-sam:adams
~alias/.qmail-anotherdomain-webmaster
```

Note that unlike sendmail, you can have two users with the same Internet user name, as long as they're in different virtual domains. In the above example, there's a webmaster@abc.com and a webmaster@anotherdomain.org.

To Handle a Virtual Domain With an /etc/aliases-like File

If you have hundreds of users for a virtual domain, you can avoid the hundreds of .qmail-*xxx* files with a small script that calls qmail's forward command.

Instead of creating individual .qmail-*xxx* files in the virtual domain master user's home directory, create a single .qmail-default file containing the following line:

```
|/usr/local/bin/qmail_db_lookup /home/master/qmail_db
```

with /home/master/qmail_db modified to reflect the home directory of your virtual domain. You can then create (or adapt from your existing /etc/aliases) the file /home/master/qmail_db, consisting of lines with the virtual domain user, a colon and the forwarding address(es). The special user name "-" indicates where mail should be forwarded for any users not explicitly listed. If the "-" user name is not provided, mail for nonexistent users will be bounced. A sample qmail_db file might look like this:

```
-:  postmaster@somewhere.com
john.smith:   john.smith@alo.com
carl.jones:   cjones@test.net
karen.quincy: kquincy
all:  john.smith@alo.com cjones@test.net kquincy
```

Note the forwarding addresses for the "-" user must be an actual address; otherwise, mail to nonexistent addresses in the virtual domain will be accepted, but not delivered to anyone.

Listing 1

A more substantial package for supporting /etc/aliases, **qmsmac**, is available with qmail. **qmsmac** supports arbitrarily deep-nested aliases and long aliases but, like sendmail, requires you to rebuild a database of aliases every time your /etc/aliases file is changed.

Conclusions

qmail appears to be a speedy and robust replacement for sendmail. We've had qmail running on our Linux Internet server for many months now without a single glitch. The additional features provided by qmail could be useful to those of you hosting several virtual domains from a single Linux box, and the simpler configuration is an added bonus.

Mike Thomas is an Internet application developer working for a consulting firm in Saskatchewan, Canada. Mike lives in Massachusetts and uses two Linux systems to telecommute 2000 miles to his job and to graduate school at the University of Regina. He can be reached by e-mail at thomas@javanet.com.

Archive Index Issue Table of Contents

Advanced search

PPPui: A Friendly GUI for PPP

Nathan Meyers

Issue #50, June 1998

Having problems setting up PPP? Mr. Meyers gives us a graphical interface to make it easy.

PPP, the Point-to-Point Protocol, is today's protocol of choice for network connectivity over a serial line. For Linux users, the tools of choice are the PPP utilities. The utilities—**pppd**, **chat** and a collection of scripts and utilities—nicely manage the connection from startup to shutdown: dialing and logging in, starting up the protocol, adding routing information and closing the connection.

What the PPP utilities do not have is a good user interface. If you use the utilities, you know the routine: execute a script called something like /etc/ppp/scripts/ppp-on, listen to your modem make noise and wait while your network connection attempts to start working. If nothing happens, check the process list. If pppd isn't running, the login attempt failed and you need to try again. If the connection goes down, you eventually figure it out by checking the process list and starting a new connection. Things improve after you've used the PPP utilities awhile—you begin to recognize the state of the connection by the sounds of disk activity.

PPPui was written out of my frustration with the utilities' low-tech interface. It's an X-based (specifically, Tcl/Tk-based) GUI that provides very simple control over a PPP connection. The goal was to provide easy startup and shutdown and useful visual feedback about the connection.

The scenario for a PPPui session looks something like this:

- Start PPPui. I've defined a window manager action, described below, that makes this a one-click operation.
- A window comes up and reports the progress of the login process. If login fails, the window disappears and you must try again.

- Once login succeeds and the PPP route is added, PPPui displays a running clock of your connect time. The clock is visible even when you iconify PPPui.
- To terminate the connection, press PPPui's “Quit” button.

PPPui is implemented as a **wish** (WIndowing SHell—part of the Tcl/Tk package) script and requires wish4.2. It does not replace the PPP utilities, but does require some very minor changes to them—discussed later in detail.

A Look at the Program

The PPPui script was developed on a Slackware system using the PPP utilities source distribution. In the following description and in the script itself, there are assumptions about locations (scripts and executables) and permissions that do not necessarily apply to other distributions. The script takes two required arguments and two optional arguments. The basic invocation takes a PPP startup command, a PPP shutdown command and optionally the name of the PPP device being opened (defaults to **ppp0**). For example:

```
PPPui /etc/ppp/scripts/ppp-on\
        /etc/ppp/scripts/ppp-off ppp0
```

The startup and shutdown commands are interpreted by /bin/sh when they are invoked and can contain multiple arguments and anything else the shell will recognize. For example:

```
PPPui /etc/ppp/scripts/ppp-on\
        '/etc/ppp/scripts/ppp-off ppp0' ppp0
```

The other optional argument, **-small**, is described in the section “Interacting with the Window Manager”.

After parsing the command line, PPPui sets the shell command to use when interpreting startup and shutdown commands. It then sets up a simple GUI containing a label, a “quit” button and a scrolled text console for displaying status information. PPPui also sets up a named pipe to the console, whose purpose is explained in the section “Changes Required in PPP Utilities”, and it initiates the startup command and captures the output. After setup, PPPui's most important job is to wait around for something to happen. Here are the things it is waiting for:

- The startup command generates output or output is received through the named pipe. The output is captured by HandleStartOutput{} or HandleFIFOInput{} (respectively) and sent to the console by calling ToConsole{}.

- The PPP connection is established. PPPui calls CheckPPPDevice{} every 1/2 second to examine the contents of /proc/net/route until a route appears for the device (the third argument to the script names the device). Once the route appears, PPPui starts a running display of the connect time, calling PostTime{} once a second to update the clock. The clock is displayed in the label at the upper-left corner of the GUI and also in the window's title. This allows the clock to be visible even when the window is iconified.
- The user asks to close the connection by pressing the “Quit” button or closing the window. PPPQuit{} is called to invoke the shutdown command. After the invocation, PPPui resumes waiting. Any output from the shutdown command is captured by HandleStopOutput{} and sent to the console.
- The startup command terminates. This is usually in response to execution of the shutdown command, but can also occur if the connection dies or the login fails. The death of the startup command results in an EOF condition detected by HandleStartOutput{}, at which point PPPui terminates and its window disappears.

Changes Required in PPP Utilities

PPPui works with the existing PPP utilities as is—*almost*. There is one required change and one optional change to effect the interface from the utilities to PPPui:

- Required: The startup script must not terminate until the connection is closed. Normal pppd behavior is to run in the background as a daemon—this does not work for use with PPPui. You need to add the **-detach** option to the invocation of pppd in the **ppp-on** script. In my version of ppp-on, based on the 2.2.0f distribution of the PPP utilities, the pppd invocation appears thus:

```
exec /usr/sbin/pppd debug lock modem crtscts\
/dev/modem 38400 asyncmap 20A0000 escape FF\
kdebug 0 -detach $LOCAL_IP:$REMOTE_IP\
noipdefault netmask $NETMASK defaultroute\
connect $DIALER_SCRIPT
```

- Optional: The chat program, which handles the dialing and login responsibilities, has the option of logging all text received from the remote modem to stderr—that is, to display what you would see if you were logging in from a terminal. If this option is enabled with the **-V** command, the resulting stderr provides a nice display of the progress of the login, which would be useful feedback on the PPPui console. Unfortunately, chat's stderr is eaten by pppd for logging purposes. So PPPui provides an alternate route to the console: a named pipe whose name it advertises through the **PPPUI_PIPE** environment variable. By adding **-V** to the chat invocation and redirecting stderr to the named pipe,

you capture the login sequence on the console. An excerpt from my version of the **ppp-on-dialer** script:

```
if [ "$PPPUI_PIPE" != "" ] ;
then exec 2>"$PPPUI_PIPE" ; fi
exec /usr/sbin/chat -V
```

Figure 1. PPPui Screen Dump

Figure 1 is a screen dump of PPPui taken ten seconds after I established a connection with Teleport, my ISP in Portland. The login sequence is captured in the console, and the clock ticks away in the upper left corner and on the title bar. If I iconify the client, the running clock appears with the icon or in the task bar (depending on your choice of window manager).

What about security and permissions? One more change to the utilities seems appropriate, and it is addressed in the sidebar, "PPP Security and Those Pesky Run-Only Scripts".

Interacting with the Window Manager

By adding a PPPui action to the window manager, you can make starting your PPP connection a one-click operation. I use **FVWM**95 as my manager; this line in my .fvwm95rc adds PPP startup to the **FvwmButtons** panel:

```
*FvwmButtons(Title 'teleport', \
Action 'Exec "Teleport Connect" \
PPPui -name "Teleport Connect" \
"/etc/ppp/scripts/ppp-on" \
"/etc/ppp/scripts/ppp-off" ppp0 &')
```

PPPui has an additional option to facilitate interaction with FVWM95. The **-small *windowname*** argument causes the following two operations to occur after the connection is established:

1. The console is unmapped and the clock and quit button repacked into a very small area.
2. The window title is changed to the window name specified as part of the option.

To support the "swallowing" function of **FvwmButtons**, the following entry is needed in .fvwm95rc:

```
*FvwmButtons(Title 'teleport', Swallow \
        (Useold Respawn) "pppui_swallow"\
        Nop, Action 'Exec PPPui -name \
        "Teleport Connect" -small pppui_swallow\
        "/etc/ppp/scripts/ppp-on" \
        "/etc/ppp/scripts/ppp-off" ppp0 &')
```

It creates an invocation of PPPui that is swallowed into the button panel after the connection is established, as shown in Figure 2. (Unfortunately, the semantics of FvwmButtons cause the button to appear in a constantly-pressed state when the connection is not active, which is a bit distracting.)

Figure 2. FvwmButtons Example

Conclusion

PPPui replaces a hard-to-use and obscure user interface with a simple, intuitive graphical interface. The code for the PPPui program and the rscript utility are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue50/2491.tgz.

Resources

Nathan Meyers is a staff consultant in Portland for a large electronics company, where he specializes in application performance of UNIX-based products. He can be reached at nmeyers@teleport.com.

Archive Index Issue Table of Contents

Advanced search

Quickly Setting Up PLIP and NFS

Loris Renggli

Issue #50, June 1998

Need to transfer files between your desktop and your laptop? Here's the easy way to do it by networking.

If you have two computers running Linux, one of which is a notebook, you are most likely tired of exchanging data between them using floppy disks. This article explains how to quickly set up these two computers so that you can use networking instead. Don't be afraid if you have no prior knowledge of networking; just follow the instructions step by step. If you have successfully installed Linux by yourself on your computer, you will be able to do this as well.

I'll start by setting up a PLIP (Parallel Line Internet Protocol) connection, which is simply a network connection through the parallel port. This port is most often used to connect a printer, and it is most probably located at the back of your computer. It has a connector with 25 holes. You will need a special cable to make this connection. Once made, you will have a full network connection allowing you to use **ftp** or **rcp** to transfer files between the computers.

Next, I'll discuss using NFS and *mounting* the disk of the notebook computer on the desktop computer. In this way, the disk of the notebook computer will appear as if it were a local disk on your desktop computer, and you can manipulate (edit, copy, etc.) your notebook files using your favorite commands.

Finally, I will show you how to access the Internet through PLIP from the notebook computer if your desktop computer has Internet access.

Assumptions and Conventions

I am using the Slackware 3.2 distribution of Linux (kernel 2.0.27), so if you have another distribution, some interpolation may be necessary—in particular, for

the kernel configuration and the location of the system files. You will need the following:

- two computers with Slackware 3.2—generic Slackware kernel—or your favorite Linux distribution
- root access on both computers
- your own account on both computers (with same UID for NFS)
- TCP/IP package installed
- parallel port **plip1** on each computer (IO 0x378, IRQ 7)

I will use the following conventions for commands:

- commands with prompt ending in **#** are issued as root
- commands with prompt ending in **>** are issued as an unprivileged user

Finally, when editing files as root, remember to *always* make a backup copy of all configuration files before you alter them during the configuration process!

Part 1. What is PLIP?

PLIP is similar to SLIP (Serial Line Internet Protocol), except that it uses a parallel cable for the connection. SLIP is used for networking over serial lines (like modems, or the serial ports of your computer, usually with 9-hole connectors). Your printer and the PLIP connection cannot both be used at the same time, since they both use the parallel port. However, our primary goal is to have a temporary connection between the two computers, and switching between the printer and the connection is quite easy. You will have to connect/disconnect the cable manually, which may involve crawling under your desk. If you do this often, you may wish to consider buying a data switch box.

Setting Up PLIP

As already mentioned, the first thing you need is a “null-printer” parallel cable, which is often sold under the name “Laplink” cable or “PC-to-PC” cable. It is cheap (about $10 US) and easy to find in any computer store. There are also instructions on how to build one yourself in the NET-2-HOWTO, but I don't think it is worth the trouble and you could end up damaging your parallel port if you make a mistake—so just buy one.

Check the Kernel

Next, check your kernel. If you are using the distribution kernel that came with your Slackware 3.2 distribution, then you're all set. (If you don't know which kernel you are using, then you probably just have the generic one.) If not, check that you have loadable module support, networking support, PLIP and printer

support as a module (no built-in printer support). If you have to recompile the kernel, then check the appropriate documentation and make sure to turn on these options:

```
CONFIG_MODULES=y
CONFIG_NET=y
CONFIG_INET=y
CONFIG_NETDEVICES=y
CONFIG_PLIP=m
CONFIG_PRINTER=m
```

Recompiling the kernel is not hard. You need to know what hardware you have and understand what all the options mean. Check the Kernel-HOWTO and the Documentation/Config.help file that comes with the kernel sources. If you have to recompile the kernel, first read this entire article, because later I will mention some additional options you may want to turn on.

With the correct options for the kernel, start the configuration, taking the following steps (as described in the next two sections):

- Check for modules in /etc/rc.d/rc.modules and update /etc/hosts.
- Write scripts to start/stop the connection.

Network Setup

You have to be root to edit the files. On both computers, in /etc/rc.d/rc.modules, comment out the line enabling printer support. (To comment out a statement in the file, just put the **#** character at the beginning of the line.) For example:

```
#/sbin/modprobe lp
```

Check that PLIP support is also commented out.

```
#/sbin/modprobe plip
```

I will load/unload the modules as needed from script files. Choose names for the two computers; I will call the desktop computer “zeus”, and the notebook “hermes”. (Hermes was the god of travel and business in Ancient Greece, and Zeus was his “boss”, being the god of the sky and master of all gods.) On both zeus and hermes, edit the file /etc/hosts and add the following two lines:

```
192.168.93.1     zeus
192.168.93.2     hermes
```

The addresses 192.168.93.xxx are safe to use; they will not conflict with existing addresses unless you already have a local network using these addresses. These IP addresses follow the convention for IP addressing: they are used only for local networks. (See NET-2-HOWTO and RFC1597 for more information.) You

could skip this step and use the numeric addresses, but it is easier to remember zeus and hermes.

Scripts

On zeus, create the following script, /usr/sbin/plip-on.sh:

```
#!/bin/sh
/sbin/modprobe -r lp
/sbin/modprobe plip
/sbin/ifconfig plip1 zeus pointopoint hermes up
/sbin/route add hermes dev plip1
```

The **modprobe** commands unload module **lp** (printer module) and load the **plip** module. (Actually, PLIP works on my system with **lp** loaded, but the available documentation says it won't work; feel free to experiment.) **ifconfig** is then run to set up the network interface **plip1**. **route** tells the computer how to find its way to the network; here, host hermes is located through the network interface **plip1**. Next, create the following script, /usr/sbin/plip-off.sh:

```
#!/bin/sh
/sbin/route del hermes
/sbin/ifconfig plip1 down
/sbin/modprobe -r plip
/sbin/modprobe lp
```

Similarly, on hermes, write the following script, /usr/sbin/plip-on.sh:

```
#!/bin/sh
/sbin/modprobe -r lp
/sbin/modprobe plip
/sbin/ifconfig plip1 hermes pointopoint zeus up
/sbin/route add zeus dev plip1
/sbin/route add default gw zeus dev plip1
```

The main difference between the plip-on.sh file on zeus is that I have swapped **zeus** and **hermes** throughout. I have also added a default route, that is, when a connection (other than to zeus) is requested to the network, I use **plip1** by default. I need this default to connect hermes to the Internet through zeus using masquerading as discussed at the end of this article; for PLIP and NFS, it is not needed. Now write the following script, /usr/sbin/plip-off.sh:

```
#!/bin/sh
/sbin/route del default
/sbin/route del zeus
/sbin/ifconfig plip1 down
/sbin/modprobe -r plip
/sbin/modprobe lp
```

Remember to change permissions (**chmod +x plip-*.sh**) on both computers to make the scripts executable. Now, you can plug in your cable and issue this command (as root):

```
# /usr/sbin/plip-on
```

on both zeus and hermes. (It does not matter on which you issue it first.) You should now have full connectivity between zeus and hermes. From hermes type:

```
hermes:~> telnet zeus
```

and log in to zeus. Congratulations. You have just set up your own private local network.

Running the Scripts as root

Working as root to run the scripts is not only annoying, it is also potentially dangerous. You could easily damage your Linux system by mistakenly removing files thus losing precious hours of sometimes difficult and tedious customization. That's why you should use the technique described in “Safely Running Programs as Root” by Phil Hughes, *Linux Journal* May 1997, and create executables named **plip-on** and **plip-off** with suid root to allow any user to start and stop the connection. All the executables do is run the scripts assuming root identity regardless of which user runs them. Example source code is available at ftp://sunsite.unc.edu/pub/Linux/docs/linux-journal/listings/issue37 in the file 2114.tgz.

Transferring Files

I now have a full network connection between zeus and hermes, so all the network software will work (TELNET, FTP, rlogin, etc.). Try to exchange files between the two computers using **ftp**. (The FTP server is turned on by default in Slackware; check the /etc/inetd.conf file and the man pages for inetd(8), ftpd(8).) Quicker even than ftp, you can use **rcp** (remote copy; see the man pages rcp(1), rlogin(1), rsh(1)). The transfer rate that I get on my systems is about 25KB per second using FTP.

Part 2. What is NFS?

NFS (Networked File System) allows you to access remote file systems of other computers through a network connection. In other words, you can manipulate files on another computer directly, as if they were files on your own computer. Your kernel must have NFS support enabled (default on Slackware), and you need to run programs (called “daemons”) which listen to the network for connection requests and act accordingly when one is received. You also need to specify which directories can be accessed and which hosts are allowed to access them.

Setting Up NFS

NFS is an even better way to access files between the two computers than using ftp or rcp. (To have NFS support enabled, the option is CONFIG_NFS_FS=y; again, this is the default with Slackware.) The setup described here allows you to consider the disk of hermes as being a disk on zeus, thereby allowing you to access all the files directly without having to log in (as with ftp) or setting rhosts access (as with rcp). Before you start, check your user identification number (UID) on both machines using the command **id**:

```
zeus:~> id
uid=401(zeusname) gid=100(users) groups=100(users)
hermes:~> id
uid=401(hermname) gid=100(users) groups=100(users)
```

I will assume that both numbers to the right of “uid=” match. (This number could be something other than 401.) If they do not match, refer to the section ahead called “If UIDs Don't Match”. Now take the following steps (described in the next three sections):

- Start the RPC (remote procedure call) daemons in the /etc/rc.d/rc.inet2 file.
- Create a list of hermes directories to be exported.
- Mount the exported directories of hermes on zeus.

RPC daemons

On hermes, check that the **rpc** daemons are launched in the /etc/rc.d/rc.inet2 file. These daemons process the network requests to handle NFS. They are launched by default in Slackware, so if you haven't changed the re.inet2 file from the original distribution, there is nothing to do, and you can skip the rest of this section.

First, type **ps a | grep rpc** to check that the daemons are running. On my system, I get this output:

```
hermes:~> ps a | grep rpc
80  ?  S     0:00 /usr/sbin/rpc.mountd
83  ?  S     0:00 /usr/sbin/rpc.nfsd
74  ?  S     0:00 /usr/sbin/rpc.portmap
```

If they are not running, edit the /etc/rc.d/rc.inet2 file (make a backup copy first) and append the lines:

```
/usr/sbin/rpc.portmap
/usr/sbin/rpc.mountd
/usr/sbin/rpc.nfsd
```

If you make any changes to the /etc/rc.d/rc.inet2 file, reboot the computer and check to be sure the daemons are now running. These commands could always be issued as root from the command line instead of rebooting.

Exporting Directories

On hermes, edit the /etc/exports file, which contains the directories you wish to make accessible from zeus. You have to choose which directories you want to export. You could just export the root directory **/**, thus exporting the whole disk; but usually you just want to access the user files located in /home, so in this example I export only /home. Add the following line to the /etc/exports file:

```
/home  zeus
```

See the exports(5) man page for the format of the file and the available options. In particular, you will be able to write from zeus onto hermes' disk; if you don't think this is a good idea, use:

```
/home  zeus(ro)
```

The option **ro** stands for “read-only”. Notice that user root cannot write on hermes' disk from zeus unless you specifically allow it using the options described in exports(5). You can add other directories, one on each line, using the same syntax.

Now tell the **nfsd** and **mountd** daemon that the exports file has changed by sending them a signal using the command:

```
hermes# killall -HUP rpc.nfsd rpc.mountd
```

Mounting the Directories

From zeus, you can check that hermes is now ready to export by issuing the command:

```
zeus# /usr/sbin/showmount -e hermes
Export list for hermes:
/home    zeus
```

Finally, on zeus, chose a “mount point”; this is just an empty directory that you will use to access the remote directory on hermes. I suggest:

```
zeus# mkdir /nfs
zeus# mkdir /nfs/hermes
```

Ready to mount? From zeus type:

```
zeus# /sbin/mount -t nfs hermes:/home /nfs/hermes
```

All the files on hermes in directory /home are now accessible from zeus. Type:

```
zeus:~> ls /nfs/hermes/hermname
```

and you will see the listing of all your files on hermes (if your account has “hermname” as user name). Once you are finished and wish to shutdown the notebook computer, unmount hermes' file system by giving:

```
zeus# /sbin/umount /nfs/hermes
```

then close the connection using the script, plip-off.sh.

I can even make things a bit more comfortable by adding the following line in the file /etc/fstab on zeus:

```
hermes:/home  /nfs/hermes  nfs  noauto   0  0
```

This command tells Linux to add to its list of file systems the directory /home on host hermes, which has to be mounted under /nfs/hermes on zeus, but not automatically (in particular, not at boot time), and that hermes has the type **nfs**. (See the man pages nfs(5), filesystems(9), fstab(4) for details.) By adding that line, the mount command is reduced to:

```
zeus# /sbin/mount /nfs/hermes
```

Finally, if you are using PLIP solely to use NFS, you could add the **mount** command at the end of the zeus script plip-on.sh and **umount** at the beginning of plip-off.sh. In this case, you must start PLIP first on hermes and shut it down first on zeus, otherwise mount cannot reach the network.

If UIDs Don't Match

If you have two different UIDs on hermes and zeus, you can still read files on zeus from hermes in directory /nfs/hermes/hermname provided the files are world readable. Thus, you can list and copy files, but you won't be able to modify them, even if the directory was exported with read and write access. This is true because zeusname and hermname are considered to be two distinct users. Linux identifies users by using their UID, not their account names.

If all you want is to copy files from hermes to zeus and it does not bother you that files are world (or group) readable on hermes, you can leave things the way they are. More likely you will wish to write files on hermes' disk as well as read them. You must choose the same UID for zeusname on zeus and hermname on hermes, and it must be *distinct* from any other UID that might already exist on both systems. It is also better to have the same identification number for the group, GID, but this is probably the case already; using Slackware, the default group for users is just “users” with GID equal to 100.

The UID is stored in the /etc/passwd file. On each line of this file is the information for a particular user beginning with the user name. The UID is the third field from the beginning of the line, fields being separated by colons (:). Once you have determined a UID that suits you, edit the /etc/passwd file and insert the proper UID. Assuming the UID 410 on zeus can be used on hermes as well and the GID is 100, there is nothing to change on zeus. Then change ownership of the files in your home directory on hermes by giving the command:

```
hermes# chown -R 410.100 hermname .
```

Make sure you issue the command from your home directory or else expect to face some problems. Have a look at the man page of **chown** to make sure you understand how it works. In the case where you have to change the UIDs on both zeus and hermes, issue the same command from your home directory on zeus.

Note: Make sure that changing ownership won't affect file access for other users. This could cause trouble if you had files or directories that you own but which you share with other users in the same group.

IP Masquerading

I now have two computers, connected using a local network with PLIP, file sharing with NFS. Sounds pretty good, doesn't it? Yes, but we want more.

If we have Internet access from zeus via a PPP connection (or an Ethernet card), we can use zeus as a gateway for hermes to have Internet access. In other words, we can use the PPP connection on zeus from hermes through the parallel cable. We will do this using IP masquerading. This is the subject of a stand-alone article (see “IP Masquerading with Linux” by Chris Kostick *Linux Journal* July 1996), so I am just going to give pointers to some relevant documentation. See Resources for the web address of the IP Masquerade HOWTO and the Firewall-HOWTO. You will have to recompile the kernel to activate masquerading. Create a /usr/sbin/ipmasq-on.sh script on zeus containing the lines:

```
#!/bin/sh
ipfwadm -F -p deny
ipfwadm -F -a m -S 192.168.93.0/24 -D 0.0.0.0/0
```

Invoke the script as root on zeus when the PLIP connection is active. I know this is not enough information, but masquerading is not the subject of this article and the HOWTO contains all the details.

Conclusion

I said in the beginning of the article that I just wanted a temporary connection between zeus and hermes; it looks like I got a little carried away and am close to making it permanent. By the way, I wrote most of this article on hermes, with the file residing on a zeus file system mounted via NFS. Yes, I think it is truly becoming permanent.

Resources

Loris Renggli is a mathematician who holds a Ph.D. in numerical analysis from the Swiss Federal Institute of Technology. For the past four years, he has been teaching and doing research in Switzerland, Finland, France and the U.S. He was last seen at Florida State University as an assistant professor in mathematics. When you will read this article, he will be elsewhere; he prefers to be reached electronically at renggli@math.fsu.edu.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Introducing the Network Information Service for Linux

Preston Brown

Issue #50, June 1998

NIS is a system for sharing system information between machines. Mr. Brown tells us how to set up and use it.

As networks of computers grow, it is increasingly important to share common system information between them to present a unified appearance to the computer user. To share files, we use NFS or Samba protocols to communicate with a file server. Mail is often kept in a central location and accessed from clients with POP3 or IMAP protocols. However, most Linux distributions aren't ready “out of the box” to share configuration file information. This doesn't have to be the case.

The Network Information Service (NIS) is a system originally invented by Sun Microsystems for distributing shared configuration files seamlessly across a network. Files such as /etc/passwrd and /etc/group become more and more difficult to maintain by hand as the number of machines grows beyond two or three. For instance, if user *foo* changes his password on one computer in a work group or cluster, he will probably expect to be able to sit down at a machine across the room and use his new password to log in. However, because the password configuration file is not shared, the reality is that he now has two passwords; the new one on the machine he just sat at and the old one on all the other machines. Each time he uses a new computer he will have to change his password. As the number of machines increases, this becomes more than merely annoying.

While schemes other than NIS exist for keeping files synchronized, they are generally “hacks” involving **rcp** and **cron** and leave much to be desired in the way of flexibility. Additionally, NIS has become a UNIX industry standard, supported by many of the commercial UNIX vendors, including IBM (AIX), Hewlett Packard (HP-UX) and of course Sun (SunOS/Solaris). As a result, when setting up Linux computers in a heterogeneous UNIX environment, NIS is often the rule rather than the exception.

Linux NIS Support

There are two major versions of NIS. The first, called NIS but historically referred to as Yellow Pages or YP, is well-supported by Linux. The second, called NIS+, is a new implementation of NIS which attempts to address security issues in the original protocol through encryption and other methods. NIS+ support in Linux is of beta quality and is readily available only in the new **glibc/libc6**, which currently applies to a very small fraction of the installed Linux base. NIS+ is best suited for very large installations: in most cases the "original" NIS is sufficient and reasonably secure when the proper precautions are taken.

Linux has included most of the components necessary to support NIS for a long time, but unfortunately distributions such as Slackware and Red Hat fail to provide adequate setup information or a complete set of tools.

Documentation has also historically been severely lacking; the NIS-HOWTO was neglected and outdated for a very long time, and it was only in November of 1997 that it finally received some major updates from one of the new NIS maintainers. In general, setting up NIS on Linux has been perceived as one of the hardest administrative tasks an administrator could undertake. I hope to change that sentiment with this article.

NIS support involves several things. To serve configuration files, one computer acts as an NIS "master". It is possible to have several NIS server machines, but machines in addition to the master are called "slaves" and are provided only for fail-safe redundancy. All of the other computers that receive information from the server computer(s) are called "clients". This article deals primarily with setting up Linux NIS clients, but describes the basics of configuring Linux to act as a server as well. A description of configuring NIS slaves and other advanced features are beyond the scope of this article. If you need more information in these areas, I suggest that you obtain the excellent book from O'Reilly, *Managing NFS and NIS*, which, although Sun-oriented, has much information applicable to Linux as well.

Setting Up an NIS Server

If you do not have an NIS master server on your network, you will need to set one up. If you do have one, you can skip this section. The package for setting up NIS servers is logically called **ypserv**, and the latest stable version at the time of this writing (February) is 1.2.6. Unfortunately, Red Hat 4.2 ships with an older version that you probably don't want to use. There is a contributed RPM in the directory /contrib/hurricane/SRPMS on the Red Hat mirror sites, and if you get the source RPM, you can rebuild it and use it. Otherwise, the URL at which the original distribution can be found is ftp://ftp.uni-paderborn.de/linux/local/yp/.

After configuring and installing the ypserv package (the default locations it specifies are fine), make sure that it will be started at boot time. The RPM will install the appropriate SYSV-style **init** scripts, but if you are using another Linux, you will have to add a command to start /usr/sbin/ypserv in the configuration file from which network daemons are started, usually /etc/rc.d/rc.inet2.

The files an NIS server makes available to its clients are not the actual local configuration files, but “compiled” versions of these that are referred to as the NIS “maps”. The process of creating NIS maps is done with a Makefile, and so I refer to the process as akin to compiling. NIS maps are in the *gdbm* UNIX database format for quick random access. The main directory where this all takes place is /var/yp.

You also have to update a few configuration files specifically for your site. If you don't want all the traditional files published via NIS to be available to clients, you will have to edit /var/yp/Makefile and remove the rules for files you wish to omit. A typical rule might look like this:

```
networks: networks.byaddr networks.byname
```

A stock Red Hat 4.2 system doesn't come with the /etc/networks file; you will get an error when running **make** if the file is not present. Comment out this line with a “#” character at the beginning. Luckily, the Makefile has a great deal of documentation included with all the different options so making changes to it is a breeze.

For security reasons you may also want to edit /etc/ypserv.conf and change a few defaults. The man page for ypserv.conf explains each of the options. The most important ones are **sunos_kludge**, which you will have to set to “yes” if you have old SunOS 4.1.x machines on your NIS network, and the “hosts” section that allows you to include and exclude certain hosts from accessing specific NIS maps on the server. The **ypserv** daemon also supports the TCP wrappers package, and this gives you another option for fine-grained access control. This is not an article on security, though, so if you are installing the server in an environment where security is of the utmost importance, read the documentation that comes with the server carefully.

You're almost done; the last thing you need to do is to create the initial build of NIS maps from the local configuration files. To do this, run **make** in /var/yp. This will create a lot of files that you will never have to alter, but these are the files that actually get served to the clients. If you get an error during the make process, chances are that one of the files make is trying to convert to an NIS map is not present. Check for its presence; if it is not there or if you don't want it to be built, comment it out of the Makefile.

If you want users to be able to change their passwords on the client machines instead of only on the server, you must run the **rpc.yppasswdd** daemon. It comes with the NIS server and has good documentation. All it has to do is run at boot time after ypserv starts up.

Now the setup is done. Reboot your machine, and when it comes back up, use **ps** to make sure ypserv and rpc.yppasswdd are running. If they are, clients should be able to connect. There are only a few final notes before moving on to client setup. First, remember that the NIS maps are *not* the same as the local configuration files, even though they are created from them. Each time you change a local file such as /etc/passwd or /etc/netgroup, you need to rebuild the NIS maps by going into /var/yp and running make. You might want to set up a **cron** job to do this once an hour in case you make some changes but forget to rebuild. Second, if you want to be able to see the NIS maps on your server, you will have to configure it as a client. Follow the steps in the next section to do so.

Setting Up an NIS Client

I will describe how to set up NIS on a Linux machine running the Red Hat 4.2 distribution. However, what I cover is generally applicable to any Linux machine running a libc5-based system with the **NYS** NIS support library included in the libc (as the case is with Red Hat), or a new libc6 system. If you are unfortunate enough not to fall into either of these cases, I suggest upgrading your Linux distributions across the board, or going out and getting a different libc at sunsite.unc.edu. (Upgrading libc is not a task for the faint-of-heart.)

The two packages required to make Red Hat Linux into an NIS client are **yp-clients-2.2** and **ypbind-3.3**. These tools start with the letters “yp” for historical reasons. If you want to get the tar.gz files, they are available at ftp://ftp.uni-paderborn.de/linux/local/yp/. Otherwise, you can use pre-packaged RPMs from any Red Hat FTP site. The **yp-clients** package is in the standard binary RPMS directory, but the **ypbind** package is only available in the “contrib/SRPMS” area. You can run NIS without ypbind, but not all features will be available. You will want those features though, so get the RPM or source code, compile it and install.

Once you have those packages installed, verify that you have the portmapper running. The portmapper enables the Remote Procedure Call (RPC) mechanism, which is used by NIS and NFS, among other things. If you are already running NFS, you don't need to worry. Otherwise, install the portmap package for your appropriate distribution; it should be included.

Now that you have the various components installed, you must set up a few configuration files to enable NIS at boot-time. First, you have to determine the

NIS domain in which the client will run. This is *not* the same thing as your DNS domain, although sometimes the name is the same. The traditional location for the NIS domain is in the file /etc/defaultdomain. This file should contain a single line with the name of your NIS domain, for example, **econ.yale.edu**. However, this file's mere presence does not set the domain name; to actually set the name you must use the **domainname** command. Usually this is done at boot time from the init scripts. On Red Hat, add the line:

```
domainname `cat /etc/defaultdomain`
```

near the top of /etc/rc.d/rc.sysinit. On non-SYSV init systems like Slackware, the appropriate file to change would most likely be /etc/rc.d/rc.inet2. The main objective is to get domainname to run “early” in the boot process, before other network services that might depend on NIS run, but after the network hardware is initialized and the IP address information is set.

Next, you have to start the ypbind daemon, which turns on the NIS client services. If you installed the ypbind RPM referenced above, the proper SYSV-style startup files should be ready to go. Otherwise, you will have to start ypbind yourself. Insert the call to /usr/sbin/ypbind right after the line initializing the NIS domain name.

Third, create the file /etc/yp.conf and include the line:

```
ypserver
```

Replace ***servername*** with the host name of your NIS master or slave server, whose host name must also be listed in /etc/hosts. You can list multiple servers on separate lines; if one host is down at boot time, the client will try the others. Failure to create this file will not disable NIS, but ypbind will broadcast a request for a server across your entire IP subnet, and this can be a major security hole. Someone can set up a renegade NIS server on your subnet and then feed you false configuration files. Better safe than sorry—create /etc/yp.conf.

That wasn't too bad, was it? Now you're ready to go. Reboot your machine, and when it comes back up, log in and do a few diagnostics. First type **domainname** without any arguments to be sure that it responds with the proper NIS domain. If not, check that the command to set the domain has run properly. Then type **ypwhich**, the command to tell you which NIS server you are talking to. If all is well, it will respond with the one you listed in the /etc/yp.conf file. Now it's time to put NIS to work.

Common NIS Commands

There are a number of useful command-line utilities, used to query the NIS system, that come with the yp-clients package. **ypwhich** tells you what NIS server you are bound to. Typing the command **ypcat *filename*** will display the contents of a file served over NIS. For example, **ypcat passwd** will display the NIS password file. Typing the command **ypmatch *key filename*** will match a "key" in the file name specified. For instance, I could do a **ypmatch pbrown passwd**, and it would return the entry for my account. These commands can be useful in shell scripts, debugging NIS setup troubles and numerous other creative ways.

Setting Up nsswitch.conf

The file used to tell your client computer which files to get from NIS and which files to get locally from the hard disk is called /etc/nsswitch.conf. This file is included with Red Hat Linux, but if you don't have a copy, you should be able to obtain a sample from a libc distribution or from the NIS-HOWTO. The file is in the same format as the one from Solaris, so if you are familiar with that one, you're a step ahead. If not, a short tutorial is useful.

The nsswitch.conf file consists of entries in the following form:

```
filename:
```

The part of the line before the colon describes the file in question, say passwd or group. The information after the colon describes how that file should be obtained. Access methods we are concerned with include **nis**, **compat** and **files**. The access methods are cumulative; if multiple methods are specified, they *add up* to create the final file. For instance, the entry

```
passwd: files nis
```

will first read the local password file, and then the NIS password file, producing a conglomeration of the two. If there are duplicate entries, the access methods coming first in the list take precedence.

Most likely, the lines all say **files nisplus nis** for the access methods. You don't have NIS+ running, so that method will be skipped and all will be well—you will end up with a combination of your local files and the NIS files, which is probably what you desire. However, you may want finer control over which entries users from the NIS file get added to your local file. This is where the **compat** access method comes in.

If you list **compat** as your sole access method, a special syntax in the passwd and group files is enabled. By default, the NIS version of the file isn't consulted; only the local file is used. To add entries from the NIS file, you put lines in the passwd/group file starting with a “+” character. For instance, if I wanted to add the single user *steveb* to my local passwd file, I would put the following line at the end of the passwd file:

```
+steveb::::::
```

The colons are place holders for fields normally included in the password file. Any information which is omitted will be filled in from the NIS file. Here, I have omitted everything but the user's name; all the other information (password, full-name field, home, shell) will be replaced automatically by NIS. Similarly, if you wanted to change steveb's shell on this computer, you might use this line as his entry:

```
+steveb::::::/usr/local/bin/newshell
```

All fields but the name and shell will be obtained via NIS. To include everyone in the password file, you would include this entry:

```
+::::::
```

The lines for /etc/group are similar. If you enable the compat access method for group, you could include individual groups by including

```
+mygroup:::
```

at the end of your group file. To include only a few users (different from those listed in the NIS map) in the group, put a line such as:

```
+mygroup:::steveb,pbrown,jrust
```

where the included users are spelled out explicitly.

Using a “-” character instead of a “+” character is also possible, and does just what you would expect—it removes the specified login(s) or group(s) from the file. The one warning is that you must include “-” lines before any other line that might include the user. For instance, if **jrust** is included in the NIS passwd file, and you want to remove him from the local computer but give everyone else access, you would add these two lines to the end of your passwd file:

```
-jrust::::::
+::::::
```

Another common situation arises where you want to exclude users from being able to login, but you want their information available in the passwd file. For this, you would add the line:

```
+::::::/bin/false
```

to the end of the passwd file, after previously including a few privileged users like System Administrators. Since users logging in will have no shell, they will be immediately kicked off, but all their information will be available on the computer via **ypcat** or the other NIS tools. One note of warning—do not put any "+" or "-" lines at the beginning of your passwd or group files or your computer will hang at boot time. This may be fixed in libc6, but with libc5 it is a problem.

The netgroup File

The **netgroup** file gives system administrators a powerful way for grouping users and hosts into distinctive groups which can then be referenced when setting up NIS access on the client computers. The netgroup file is maintained on the NIS master server in /etc/netgroup. It consists of the group name followed by one or more entries in the following format:

```
(
```

For instance, you might have a line like this:

```
sysadmins (,pbrown,) (,kbrown,) (,vladdy,) (,ieong,)
```

This would place the four users listed into the netgroup **sysadmins**. You can then use this special group in files such as /etc/passwd on the client using the "+/-" notation. For instance, to allow the sysadmins group access to a particular NIS client computer, but no one else, you would use an entry at the end of the passwd file like:

```
+@sysadmins
+:::::/bin/false
```

Note that the netgroup name is preceded by the "@" character so that NIS knows that this is a netgroup you are referring to and not a normal user name.

You can use netgroups for all sorts of creative things. Create a netgroup with dangerous users listed in it and use a "-" line in the passwd file to disallow them, but include everyone else. Create netgroups for all the "semi-private" machines you have on your network and list the valid users for those machines. Include them in your passwd file to give just those users access. The possibilities are almost endless.

Additional Details

While for the most part, the fact that you are running NIS instead of using traditional configuration files will be completely transparent, there are a few cases where this is not true. The most glaring of these is that you cannot use the passwd command to change your password on the client machines. If you do so, the entire NIS passwd map will be appended to the local /etc/passwd file

each time you use the command, which is certainly not the intended effect. This may have changed with libc6, but it is definitely a problem with the stock utilities that come with Red Hat 4.2. However, all hope is not lost; there is a solution, and it is called **yppasswd**. This utility acts just like passwd, but makes a call to the NIS server to do the actual change instead of trying to change things locally. The NIS server must support this by running the **yppasswdd** daemon. In its absence, you will have to tell users to log in to the NIS server to change their password, which is only a minor inconvenience.

If you run into troubles while setting up either the server or clients, the most likely source of your problems is the /etc/nsswitch.conf file. Make sure that each line is creating the behavior you intend, and when in doubt refer to the man page. Also, check the “+/-” syntax in your passwd and group files and make sure it follows the proper notation. Minor typos can have wide-ranging effects, so proofread carefully.

Unfortunately, I cannot hope to cover all the details of setting up a complex distributed network environment. You may well have unique problems or concerns that haven't been addressed here. If this is the case, I highly recommend the O'Reilly book on NIS (and NFS) that was previously mentioned. If all of these roads lead to nowhere, try the Linux, and especially the Sun, newsgroups on Usenet. There is a good chance someone else has dealt with your problem before.

Good luck setting up NIS on your Linux network. The couple of hours you initially invest will save you days of headaches in the long run.

Preston Brown is graduating from Yale University this spring with a B.S. in Computer Science. He has been the System Administrator for the Yale University Economics Department for over three years and is intimately familiar with the joys of setting up NIS. You can contact him with your questions and comments at preston.brown@yale.edu.

Archive Index Issue Table of Contents

Advanced search

Getting in the Fast Lane

Michael Hughes

Issue #50, June 1998

Here's how to set up a broadband connection for your home or office LAN.

With the arrival of technologies such as cable modems and xDSL (digital subscriber line) technologies, individuals can now have a moderate to extremely fast constant connection to the Internet, instead of the usual analog telephone-line-based modem. Such new technologies use different hardware and sometimes, as in the case of cable, a different transmission medium.

In this article, I will describe how to set up a cable modem with a local area network (LAN) to provide a fast connection to the outside world. However, you don't need to have a cable modem to benefit from this article, just a fast and constant/semi-constant connection to the Internet, such as ISDN or Frame Relay. The setup procedures are almost the same, with the exception that your hardware may require special drivers. Contact the manufacturer for details on support.

Cable Modems

There are three types of cable modems on the market today. One is the static type, which is always up and connected. This type is generally expensive and has a static IP address. It also offers two-way communications for downloading and uploading as well. This is the best type of modem to run a web, FTP or TELNET server. The second type of modem is the dynamic cable modem, which also offers two-way communications, but has a login procedure, where you are assigned a dynamic IP address each time you wish to use it. Such modems use a DHCP client to obtain a dynamic IP address from the headend server. The third and cheapest type of modem on the market requires that you use a telephone line to upload information, such as web page requests, and use the cable to download information, such as web pages. This type of service is generally inexpensive, but suffers from extremely high latency. This isn't the

type of modem to set up a server on, since the upstream connection is so slow. However, this type has blazingly fast download speeds.

This article will focus on the first type of modem, since those are the most common. There are excellent resources on-line for all types of modems. An excellent page for the dynamic modem is http://home.neo.lrun.com/rrlinux. This page has links to software needed such as DHCP clients and a login program for RoadRunner modem customers. A good resource on how to correctly set up DHCP for a cable modem can be found at http://elycion.geology.ualberta.ca/~dkimmel/cablemodem.html. The third type of modem, a cable modem that uses a telephone line to upload data, is pretty much controlled with proprietary software and hardware. These solutions are in place because of the high cost of outfitting existing cable plants for two-way communications, which is a basic part of the cable modem theory. Such solutions require hackish, kludgey software solutions on the Linux side. A blend of DHCP and PPP must be used to get the routing to work correctly, but after it's done, it is ultra fast. Consult the PPP HOWTO for help in getting your analog modem correctly configured. For the cable modem part, follow the instructions shown here or on the aforementioned web pages. The DHCP should work in the same manner.

With coaxial-cable-based Internet access there is no signal degradation for many miles. At this time, cable is the favored high-speed, low-cost connect method for homes and small to medium-sized businesses due to reliability factors. Cable Internet access availability is spreading to more areas each day.

Cable modem packages generally include a modem lease or purchase fee along with a service fee. Leases usually cost between 15 and 20 dollars, and service can cost from 35 to 70 dollars a month, depending on location. Speed varies from service to service. For example, @Home and RoadRunner offer speeds that are extremely fast—equal to multiple T1 performance in uploading and downloading. However, services such as GTE charge twice as much and have speeds that compete only with double ISDN, which operates at a transmission speed of 128 kilobits per second, which is roughly 15 to 16 kilobytes per second. It all depends on your provider.

xDSL Technology

xDSL Technology uses the same copper wires that have been in place for decades with hardware on both ends to accelerate data transfer. One drawback to this method is repeaters and/or amplifiers must be used along the route to prevent signal degradation. xDSL as yet remains untested and extremely expensive in terms of service and hardware. In fact, xDSL technology is still considered to be in the testing phases at this time (February), so I will focus on cable technology in this article.

xDSL technology, the telephone company's answer to cable modem technology, will probably be more expensive. There isn't an average price, since the whole technology is in its early stages. You can expect to pay anywhere from cable modem prices to several hundred dollars a month to get this service. However, the speeds of the service are phenomenal for the transmission medium. The speed you get depends on your telephone company and location.

With these two new technologies, how is the Linux user supposed to exploit them to their full potential? Easy. With a few simple pieces of software, you can have a full in-house network with a simple firewall, capable of World Wide Web browsing, Internet relay chatting and connecting to the outside world via FTP from each of the computers. Software such as CU-SeeMe, a video conferencing program, have IP Masquerading modules in order to work in this environment. Note that IP Masquerading does not allow for incoming connections to the client machines, so the client machines are truly client machines.

Setup

All of the newer Linux kernels include support for IP masquerading, which allows a group of computers on a network to access the Internet using a specified computer's Internet address. All connections from the computers must go through a specified IP masquerading “host”, or server. This server functions as the gateway machine and can be used as a DNS machine, if you set up a DNS server. With **route** and **ipfwadm**, you can set up a simple but effective packet routing scheme to deliver packets to the appropriate client machines. The prerequisites for such a setup are as follows:

- One Linux server
- Client machine(s) running a network-capable OS
- One or two Ethernet cards, depending on the type of your router
- The standard Linux network package; see your distribution documentation for details.

Ethernet Cards

I am assuming that you have Ethernet support for your client machines already. If your client machines are Linux and you haven't yet set up the Ethernet cards, read the documentation and do it—all of the drivers are most likely already working.

For the server, the first step of setting up a network is setting up the Ethernet hardware. This is relatively simple: recompile the kernel or compile a module to include support for your card or cards. In the case that your card is supported by the Linux kernel, as root compile support into the kernel by typing

make menuconfig at the top of the Linux source tree and use the menus to configure support. More information on how to recompile a kernel is available at the Linux Kernel HOWTO, located at http://sunsite.unc.edu/linux/HOWTO/Kernel-HOWTO.html.

In the kernel setup program, under “Code Maturity Level Options”, check the “Prompt for development and/or incomplete code/drivers” box, so that you will be given the option of using IP masquerading. Also, in the “Networking Options” section, check the following: Network Firewalls, Network aliasing, TCP/IP networking, IP forwarding/gatewaying, IP multicasting, IP firewalling, IP accounting, IP masquerading and IP tunneling. Although IP masquerading is experimental, it is fairly stable and must be included.

If your card (such as the EtherExpress Pro 10 PCI card from Intel) isn't supported in the kernel or support is broken, you can download and make a module for your card. A great resource for Ethernet card information on Linux is at the Linux Ethernet HOWTO, at http://sunsite.unc.edu/linux/HOWTO/Ethernet-HOWTO.html. At the Ethernet HOWTO, you should find complete information about your card and how to use it under Linux. Another great resource is Donald Becker's Ethernet drivers page, found at http://cesdis.gsfc.nasa.gov/pub/linux/linux.html. This page has drivers (many written by Mr. Becker) for many cards, including some that are in alpha stage. Be aware that many of the alpha drivers are perfectly usable and many are completely unusable. To find out, check the Ethernet HOWTO for support status. You can also read the actual source of the module, which should include instruction on installation and compilation near the top or bottom. Install the Ethernet card module into the /lib/modules/2.0.xx/net file and put the following lines into one of your startup scripts:

```
depmod -a
modprobe drivername
```

Note that the **modprobe drivername** does not include the “.o” at the end of the file name. It isn't necessary, so you shouldn't put it in; modprobe knows how to handle the loading of the module. To see if you've loaded the module into memory, type **lsmod** at the prompt. If you see a listing for your card, the module is loaded. Troubleshooting steps can also be found at the Linux Ethernet HOWTO. It's an excellent resource that should not be missed when setting up Ethernet cards.

If your cable modem or other high bandwidth device doesn't support being plugged into a hub or coax network, the simple solution is to buy a cheap NE2000 clone for the device and keep it separate from the other parts of the network. Yes, that's right, you'll have two Ethernet cards in your server computer. The number one problem concerning multiple Ethernet card

support is the order in which the cards get detected. This is important, since Linux addresses the Ethernet cards in numerical order, depending on the order detected during the boot process. If you know the IRQs or the I/O addresses of your Ethernet cards (it may be settable on board or via software), you can add this line to the top of your lilo.conf file:

```
append = "ether=irq,ioadd,eth0 ether=irq,ioadd,eth1"
```

This line tells the kernel which Ethernet devices to assign to which I/O or irq combinations. For example, if you have a 3Com 3c509b on irq 10 and memory address 0x300 and you wish that card to be eth0, you add this line to the very top of your lilo.conf file:

```
append = "ether=10,0x300,eth0"
```

For additional Ethernet cards, you just add another **ether=*x*,*x*,eth*x*** after the first one inside the append quotes, as shown in the previous example. This is the easiest method of getting the kernel to assign the proper devices to the correct cards. All modern Ethernet cards come with software or jumpers that let you set the irq and memory addresses. If they don't, look in your computer's BIOS or, if you have another OS such as Win95, look in the system settings for the mapped address. To check that the Ethernet card was properly detected, simply type **cat /proc/interrupts** and see if your card is listed there.

Configuration

Now that you have your card or cards set up, go ahead and boot into Linux. First, login or **su** as root and run the command **ifconfig**. You'll get a few paragraphs of information, stating the status of your network interfaces. At this point, your Ethernet interfaces (eth0, eth1) will not be listed, since you haven't configured them yet. The only interfaces listed should be the loop back interface, and anything else you have already set up.

Interfaces and Routing

What we wish to do now is set up each interface. In the case of a single Ethernet card system, issue the following command:

```
ifconfig eth0
```

replacing ***x.x.x.x*** with your specified IP address. This number is provided by your ISP (Internet service provider). Also change the **eth0** to whichever interface you wish the address to be mapped to. Now, run ifconfig as root. You will see the eth0 interface listed, with all the card details and transmission statistics. If you have a second card, issue the same command, this time with **eth1** instead of **eth0** and the internal network IP address. For your internal network, the

addresses should be in the form of 192.168.0.0, with 192.168.1.1 being the machine that is going to host the connection. In other words, all your other machines should be assigned 192.168.1.1, 192.168.1.2, 192.168.1.3, etc. These IP addresses are not publicly routed on the Internet and should not interfere with the outside world.

With the interfaces set up, it's time to set up routing. This may sound complicated, but it is quite easy once you are familiar with the route command. This command controls the flow of data between all network interfaces. The route man page gives complete details of all the intricacies of this command. For now, use this series of commands to configure routing:

```
route add
route add default gw
route add -net 192.168.1.0 eth1
```

Replace the *gateway_address* flag with your actual gateway machine address, also provided by your ISP. The first two commands tell the machine that the host *gateway_address* can be accessed directly via the eth0 interface. The third command says that the default route (0.0.0.0, any machine) should be accessed through the gateway *gateway_address*. The last line indicates that any machine in network 192.168.1.0 can be accessed through the interface eth1. Put these three lines and the ifconfig line above into the startup script, usually found in /etc/rc.d for Slackware or /etc/rc.d/rc.init for Red Hat. Check your documentation for your distribution.

Now set up DNS resolutions by editing the /etc/resolv.conf file to include the following lines:

```
domain isp.com
nameserver
nameserver
```

Replace isp.com with your ISP's domain, and replace *x.x.x.x* with your ISP's primary name server, and *y.y.y.y* with your ISP's secondary name server. If you don't have a secondary name server, don't worry, only one is actually needed. After you've added these lines, save the files and reboot.

When your computer is back on-line, you will be able to use your cable modem on the host machine to execute the regular Internet functions such as FTP, TELNET and visiting the WWW.

IP Masquerading

To effectively share bandwidth between computers without actual IP addresses for each computer, use internal IP addresses as discussed above. The masquerading server forwards packets from each of the client machines to the

Internet and relays the packets back to the client machines. This is done quite efficiently, with little noticeable load on the server. A tool called **ipfwadm** is used to set up "rules" for IP forwarding and denying. The following commands should also be added to one of your startup scripts (see Listing 1), after the ifconfig and route sections:

```
ipfwadm -F -p deny
ipfwadm -F -a M -S 192.168.1.0/24 -D 0.0.0.0/0
```

The first command tells ipfwadm to change the policy for IP firewalling to **deny**. The second command is a little more complicated; it instructs ipfwadm to append the commands that follow, which in this case are the **M**, **-S** and **-D** flags. The **M** adds a masquerade rule, which states that all packets with a source address of **192.168.1.0** and a destination address of **0.0.0.0** (which basically means any host machine) are accepted. The **/24** specifies the number of set bits in the netmask. Remember, in binary, you can only have a set or unset bit, and in netmasks, the value is always 255 or 11111111 in binary. You can also replace the 24 with the real netmask, which in this case would be 255.255.255.0. The zero in the **-D** rule just means that any netmask is allowed. The man page for ipfwadm for more details.

At this point, it is a good idea to restart, run all the scripts and load all the modules. If you don't want to bring the machine down, you can re-run the startup scripts and hope for the best.

Setting up the Clients

The hard part is now over, and it's time to set up the client machines which will be using IP masquerading to gain access to the Internet. In most major operating systems, the values for netmask, IP address, gateway and DNS server are required to effectively use the Internet. Also, routes must be specified in UNIX machines. Routing setup in Win95 is more transparent, although there is a route command.

On a Win95 or Macintosh machine, the first step is to install the network hardware which is probably an Ethernet card. These platforms are well supported, and the hardware should come with full documentation and software for installation. Read the operating system documentation on how to set network settings. Once you know how to set up the network, use 192.168.1.*x* for the IP address, where *x* is less than 255 but greater than 1. Don't assign identical network addresses to two computers on the same network. For the Gateway address, use 192.168.1.1 which will correspond to your server machine if you've followed my examples. For the DNS server search order, or DNS servers, enter the same DNS servers used for the server machine. Or, if you have a DNS server running on the server machine, you can

specify it as the DNS machine. Don't enable any funky WINS settings; set the interface to be the default. In Win95, this can be found under the “Advanced” tab. In other operating systems, you should be able to specify it with a command such as route. On Macintosh systems, you shouldn't have to worry about this, unless you have multiple Ethernet cards, which isn't likely. Also, on Macintosh systems, the Gateway address may be referred to as the “Router Address”. Treat this the same as the “Gateway address” term.

Your operating system may be different in setup, but the values you use are the same universally. Read your operating system documentation for more information on how to set these values.

Wrapping Up

Now is the time to test your setup, if you haven't already. Make sure your server machine is running, and all software is configured properly. Then, turn on a client machine and type **ping 192.168.1.1** at the command prompt or in the **Run** window in Win95. You should get a bunch of numbers every second or so. This means that the network is alive and kicking. Press **CTRL-C** to stop the **ping** command from pinging. Next, open up a piece of client software such as an FTP program or a web browser and bring up your favorite WWW site. If the site appears, your setup is working fine. If not, then you should go over the settings and try again. Remember, it may take a bit longer for the web site to appear through IP Masquerading than through a regular connection.

If you would like to discuss any of this with me, please feel free to e-mail me.

Credits and Resources

Mike Hughes is a high school student living in Thousand Oaks, California. He enjoys using Linux, snowboarding and traveling. His web site, detailing the slow cable modem service provided by GTE, can be found at www.psilord.com/. He can be reached via e-mail at wxh@gte.net.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The Coda Distributed File System

Peter J. Braam

Issue #50, June 1998

Carnegie Mellon University has developed an exciting file system. Mr. Braam, one of the developers, tells us all about it.

The Coda distributed file system is a state-of-the-art experimental file system developed in the group of M. Satyanarayanan at Carnegie Mellon University (CMU). Numerous people contributed to Coda, which now incorporates many features not found in other systems:

1. **Mobile Computing**:
   - disconnected operation for mobile clients
   - reintegration of data from disconnected clients
   - bandwidth adaptation
2. **Failure Resilience**:
   - read/write replication servers
   - resolution of server/server conflicts
   - handles network failures which partition the servers
   - handles disconnection of client's client
3. **Performance and scalability**:
   - client-side persistent caching of files, directories and attributes for high performance
   - write-back caching
4. **Security**:
   - Kerberos-like authentication
   - access control lists (ACLs)
5. **Well-defined semantics of sharing**
6. **Freely available source code**

Figure 1: Coda Logo (Illustration by Gaich Muramatsu)

Distributed File Systems

A distributed file system stores files on one or more computers called servers and makes them accessible to other computers called clients, where they appear as normal files. There are several advantages to using file servers: the files are more widely available since many computers can access the servers, and sharing the files from a single location is easier than distributing copies of files to individual clients. Backups and safety of the information are easier to arrange since only the servers need to be backed up. The servers can provide large storage space, which might be costly or impractical to supply to every client. The usefulness of a distributed file system becomes clear when considering a group of employees sharing documents; however, more is possible. For example, sharing application software is an equally good candidate. In both cases system administration becomes easier.

There are many problems facing the design of a good distributed file system. Transporting many files over the Net can easily create sluggish performance and latency; network bottlenecks and server overload can result. The security of data is another important issue: how can we be sure that a client is really authorized to have access to information and how can we prevent data being sniffed off the network? Two further problems facing the design are related to failures. Often, client computers are more reliable than the network connecting them, and network failures can render a client useless. Similarly, a server failure can be very unpleasant, since it can disable all clients from accessing crucial information. The Coda project has paid attention to many of these issues and implemented them as a research prototype.

Figure 2. Servers Control Security (Illustration by Gaich Muramatsu)

Coda was originally implemented on Mach 2.6 and has recently been ported to Linux, NetBSD and FreeBSD. Michael Callahan ported a large portion of Coda to Windows 95, and we are studying Windows NT to understand the feasibility of porting Coda to NT. Currently, our efforts are on ports and on making the system more robust. A few new features are being implemented (write-back caching and cells for example), and in several areas, components of Coda are being reorganized. We have already received very generous help from users on the Net, and we hope that this will continue. Perhaps Coda can become a popular, widely used and freely available distributed file system.

Coda on a Client

If Coda is running on a client, which we shall take to be a Linux workstation, typing **mount** will show a file system—of type “Coda”--mounted under /coda. All the files, which any of the servers may provide to the client, are available under this directory, and all clients see the same name space. A client connects to

“Coda” and not to individual servers, which come into play invisibly. This is quite different from mounting NFS file systems which is done on a per server, per export basis. In the most common Windows systems (Novell and Microsoft's CIFS) as well as with Appleshare on the Macintosh, files are also mounted per volume. Yet the global name space is not new. The Andrew file system, Coda's predecessor, pioneered the idea and stored all files under /afs. Similarly, the distributed file system DFS/DCE from OSF mounts its files under one directory. Microsoft's new distributed file system (dfs) provides glue to put all server shares in a single file tree, similar to the glue provided by auto-mount daemons and yellow pages on UNIX. Why is a single mount point advantageous? It means that all clients can be configured identically, and users will always see the same file tree. For large installations this is essential. With NFS, the client needs an up-to-date list of servers and their exported directories in /etc/fstab, while in Coda a client merely needs to know where to find the Coda root directory /coda. When new servers or shares are added, the client will discover these automatically in the /coda tree.

To understand how Coda can operate when the network connections to the server have been severed, let's analyze a simple file system operation. Suppose we type:

```
cat /coda/tmp/foo
```

to display the contents of a Coda file. What actually happens? The **cat** program will make a few system calls in relation to the file. A system call is an operation through which a program asks the kernel for service. For example, when opening the file the kernel will want to do a lookup operation to find the inode of the file and return a file handle associated with the file to the program. The inode contains the information to access the data in the file and is used by the kernel; the file handle is for the opening program. The open call enters the virtual file system (VFS) in the kernel, and when it is realized that the request is for a file in the /coda file system, it is handed to the Coda file system module in the kernel. Coda is a fairly minimalistic file-system module: it keeps a cache of recently answered requests from the VFS, but otherwise passes the request on to the Coda cache manager, called Venus. Venus will check the client disk cache for tmp/foo, and in case of a cache miss, it contacts the servers to ask for tmp/foo. When the file has been located, Venus responds to the kernel, which in turn returns the calling program from the system call. Schematically we have the image shown in Figure 3.

Figure 3. Client/Venus/Vice

The figure shows how a user program asks for service from the kernel through a system call. The kernel passes it up to Venus, by allowing Venus to read the request from the character device /dev/cfs0. Venus tries to answer the request,

by looking in its cache, asking servers or possibly by declaring disconnection and servicing it in disconnected mode. Disconnected mode kicks in when there is no network connection to any server which has the files. Typically this happens for laptops when taken off the network or during network failures. If servers fail, disconnected operation can also come into action.

When the kernel passes the open request to Venus for the first time, Venus fetches the entire file from the servers, using remote procedure calls to reach the servers. It then stores the file as a container file in the cache area (currently /usr/coda/venus.cache/). The file is now an ordinary file on the local disk, and read/write operations to the file do not reach Venus but are (almost) entirely handled by the local file system (EXT2 for Linux). Coda read/write operations take place at the same speed as those to local files. If the file is opened a second time, it will not be fetched from the servers again, but the local copy will be available for use immediately. Directory files (remember, a directory is just a file) as well as all the attributes (ownership, permissions and size) are all cached by Venus, and Venus allows operations to proceed without contacting the server if the files are present in the cache. If the file has been modified and it is closed, Venus updates the servers by sending the new file. Other operations which modify the file system, such as making directories, removing files or directories and creating or removing (symbolic) links are propagated to the servers also.

So we see that Coda caches all the information it needs on the client, and only informs the server of updates made to the file system. Studies have confirmed that modifications are quite rare compared to “read only” access to files, hence we have gone a long way towards eliminating client-server communication. These mechanisms to aggressively cache data were implemented in AFS and DFS, but most other systems have more rudimentary caching. We will see later how Coda keeps files consistent, but first pursue what else one needs to support disconnected operation.

From Caching to Disconnected Operation

The origin of disconnected operation in Coda lies in one of the original research aims of the project: to provide a file system with resilience to network failures. AFS, which supported thousands of clients in the late 80s on the CMU campus had become so large that network outages and server failures occurred somewhere almost every day. This was a nuisance. Coda also turned out to be a well-timed effort because of the rapid advent of mobile clients (viz. laptops). Coda's support for failing networks and servers equally applied to mobile clients.

We saw in the previous section that Coda caches all information needed to provide access to the data. When updates to the file system are made, these

need to be propagated to the server. In normal *connected* mode, such updates are propagated synchronously to the server, i.e., when the update is complete on the client it has also been made on the server. If a server is unavailable or if the network connections between client and server fail, such an operation will incur a time-out error and fail. Sometimes, nothing can be done. For example, trying to fetch a file, which is not in the cache, from the servers is impossible without a network connection. In such cases, the error must be reported to the calling program. However, often the time-out can be handled gracefully as follows.

To support disconnected computers or to operate in the presence of network failures, Venus will not report failure(s) to the user when an update incurs a time-out. Instead, Venus realizes that the server(s) in question are unavailable and that the update should be *logged* on the client. During disconnection, all updates are stored in the CML, the client modification log, which is frequently flushed to disk. The user doesn't notice anything when Coda switches to disconnected mode. Upon re-connection to the servers, Venus will reintegrate the CML: it asks the server to replay the file system updates on the server, thereby bringing the server up to date. Additionally the CML is optimized—for example, it cancels out if a file is first created and then removed.

There are two other issues of profound importance to disconnected operation. First, there is the concept of *hoarding* files. Since Venus cannot serve a cache miss during a disconnection, it would be nice if it kept important files in the cache up to date, by frequently asking the server to send the latest updates. Such important files are in the user's hoard database which can be automatically constructed by “spying” on the user's file access. Updating the hoarded files is called a hoard walk. In practice, our laptops hoard enormous amounts of system software, such as the X11 Window System binaries and libraries, or Wabi and Microsoft Office. Since a file is a file, legacy applications run just fine.

Figure 4. Hoarded Files are “sticky” in the cache. (Illustration by Gaich Muramatsu)

The second issue is that during reintegration it may appear that during the disconnection another client has modified the file too, and has shipped it to the server. This is called a local/global conflict (viz. Client/Server) which needs repair. Repairs can sometimes be done automatically by application-specific resolvers (which know that one client inserting an appointment into a calendar file for Monday and another client inserting one for Tuesday have not created an irresolvable conflict). Sometimes, but quite infrequently, human intervention is needed to repair the conflict.

On Friday one leaves the office with a good deal of source code hoarded on the laptop. After hacking in one's mountain cabin, the harsh return to the office on Monday (10 days later of course) starts with a re-integration of the updates made during the weekend. Mobile computing is born.

Figure 5. Failure Resilience Methods

Volumes, Servers and Server Replication

In most network file systems, the servers enjoy a standard file structure and export a directory to clients. Such a directory of files on the server can be mounted on the client and is called a network share in Windows jargon and a network file system in the UNIX world. For most of these systems it is not practical to mount further distributed volumes inside the already mounted network volumes. Extreme care and thought goes into the server layout of partitions, directories and shares. Coda's (and AFS's) organization differs substantially.

Files on Coda servers are not stored in traditional file systems. Partitions on the Coda server workstations can be made available to the file server. These partitions will contain files which are grouped into volumes. Each volume has a directory structure like a file system, i.e., a root directory for the volume and a tree below it. A volume is on the whole much smaller than a partition, but much larger than a single directory and is a logical unit of files. For example, a user's home directory would normally be a single Coda volume and similarly the Coda sources would reside in a single volume. Typically a single server would have some hundreds of volumes, perhaps with an average size approximately 10MB. A volume is a manageable amount of file data which is a very natural unit from the perspective of system administration and has proven to be quite flexible.

Coda holds volume and directory information, access control lists and file attribute information in raw partitions. These are accessed through a log-based recoverable virtual memory package (RVM) for speed and consistency. Only file data resides in the files in server partitions. RVM has built-in support for transactions—this means that in case of a server crash, the system can be restored to a consistent state without much effort.

A volume has a name and an ID, and it is possible to mount a volume anywhere under /coda. For example, to mount the volume u.braam on /coda/usr/braam, issue the command:

```
cfs makemount u.braam /coda/usr/braam
```

Coda does not allow mount points to be existing directories; instead, it will create a new directory as part of the mount process. This eliminates the

confusion that can arise in mounting UNIX file systems on top of existing directories. While it seems quite similar to the Macintosh and Windows traditions of creating a “network drive and volumes”, the crucial difference is that the mount point is invisible to the client: it appears as an ordinary directory under /coda. A single volume enjoys the privilege of being the root volume; it is the volume which is mounted on /coda at startup time.

Coda identifies a file by a triple of 32-bit integers called a Fid: it consists of a VolumeId, a VnodeId and a Uniquifier. The VolumeId identifies the volume in which the file resides. The VnodeId is the “inode” number of the file, and the uniquifiers are needed for resolution. The Fid is unique in a cluster of Coda servers.

Coda has read/write replication servers, i.e., a group of servers can hand out file data to clients, and generally updates are made to all servers in this group. The advantage of this is higher availability of data: if one server fails, others take over without a client noticing the failure. Volumes can be stored on a group of servers called the VSG (Volume Storage Group).

For replicated volumes, the VolumeId is a replicated VolumeId. The replicated volume ID brings together a Volume Storage Group and a local volume on each of the members.

- The VSG is a list of servers which hold a copy of the replicated volume.
- The local volume for each server defines a partition and local volume ID holding the files and meta-data on that server.

When Venus wishes to access an object on the servers, it first needs to find the VolumeInfo for the volume containing the file. This information contains the list of servers and the local volume IDs on each server by which the volume is known. For files, the communication with the servers in a VSG is “read-one, write-many”; that is, read the file from a single server in the VSG and propagate updates to all of the available VSG members, the AVSG. Coda can employ multicast RPCs, and hence the write-many updates are not a severe performance penalty.

The overhead of first having to fetch volume information is deceptive too. While there is a onetime lookup for volume information, subsequent file access enjoys much shorter path traversals, since the root of the volume is much nearer than is common in mounting large directories.

Server replication, like disconnected operation, has two cousins who need introduction: resolution and repair. Some servers in the VSG can become partitioned from others through network or server failures. In this case, the

AVSG for certain objects will be strictly smaller than the VSG. Updates cannot be propagated to all servers, but only to the members of the AVSG, thereby introducing global (viz. server/server) conflicts.

Figure 6. AVSG vs. VSG (Illustration by Gaich Muramatsu)

Before fetching an object or its attributes, Venus will request the version stamps from all available servers. If it detects that some servers do not have the latest copy of files, it initiates a resolution process which tries to automatically resolve the differences. If this fails, a user must repair manually. The resolution, though initiated by the client, is handled entirely by the servers.

Replication servers and resolution are marvelous. We have suffered disk failures from time to time in some of our servers. To repair the server, all that needs to be done is to put in a new drive and tell Coda: resolve it. The resolution system brings the new disk up to date with respect to other servers.

Coda in Action

Coda is in constant active use at CMU. Several dozen clients use it for development work (of Coda), as a general purpose file system and for specific disconnected applications. The following two scenarios have exploited the features of Coda very successfully. We have purchased a number of licenses for Wabi and Windows software. Wabi allows people to run MS PowerPoint. We have stored Wabi and Windows 3.1 including MS Office in Coda and it is shared by our clients. Of course .ini files with preferences are particular to a given user, but most libraries and applications are not. Through hoarding we continue to use the software on disconnected laptop computers for presentations. This is frequently done at conferences.

Over the years of its use we have not lost user data. Sometimes disks in our servers have failed, but since all of our volumes are replicated, we replaced the disk with an empty one and asked the resolution mechanism to update the repaired server. All one needs to do for this is to type **ls -lR** in the affected file tree when the new disk is in place. The absence of the file on the repaired server will be noticed, and resolution will transport the files from the good servers to the newly repaired one.

There are a number of compelling future applications where Coda could provide significant benefits.

1. FTP mirror sites should be Coda clients. As an example, let's take ftp.redhat.com, which has many mirrors. Each mirror activates a Perl script, which walks the entire tree at Red Hat to see what has been updated and fetches it—regardless of whether it is needed at the mirror.

Contrast this with Red Hat storing their ftp area in Coda. Mirror sites should all become Coda clients too, but only Red Hat would have write permission. When Red Hat updates a package, the Coda servers notify the mirror sites that the file has changed. The mirror sites will fetch this package, but only the next time someone tries to fetch this package.

2. WWW replication servers should be Coda clients. Many ISPs are struggling with a few WWW replication servers. They have too much access to use just a single http server. Using NFS to share the documents to be served has proven problematic due to performance problems, so manual copying of files to the individual servers is frequently done. Coda could come to the rescue since each server could be a Coda client and hold the data in its cache. This provides access at local disk speeds. Combine this with clients of the ISP who update their web information off-line and we have a good application for mobile clients too.
3. Network computers could exploit Coda as a cache to dramatically improve performance. Updates to the network computer would automatically be made as they become available on servers, and for the most part the computer would operate without network traffic, even after restarts.

Our current efforts are mostly to improve the quality of Coda. The rough edges, which inevitably come with research systems, are slowly being smoothed out. Write-back caching will be added in order for Coda to operate much faster. The disconnected operation is an extreme form of write-back caching, and we are leveraging these mechanisms for write-back caching during connected operation. Kerberos support is being added. The networking protocols supporting Coda are making this easily possible. We would like to have cells which will allow clients to connect to more than a single Coda cluster simultaneously. Further ports will hopefully allow many systems to use Coda.

Getting Coda

Coda is available by FTP from ftp.coda.cs.cmu.edu. You will find RPM packages for Linux as well as tar files of the source. Kernel support for Coda will come with the Linux 2.2 kernels. On the WWW site http://www.coda.cs.cmu.edu/, you will find additional resources such as mailing lists, manuals and research papers.

Peter adores his wife Anne, and together they love Alaska with its mountains, wildlife and a halfway acceptable population density. Nothing is better than

having a moose on your porch there or camping on a not too scary glacier. Until March 1997 Peter was a faculty member in the Mathematical Institute at Oxford. In the summer of 1995 Peter became president of Stelias Computing Inc. which assembled the InfoMagic Workgroup Server. Dabblings in Mach and the GNU Hurd evolved into porting Coda to Linux. E-mails about this with Satya, the visionary leader of the Coda and Odyssey projects, led to a visit to Carnegie Mellon University in late 1996 and eventually to him joining the Computer Science faculty. He is now leading the Coda project. He can be reached at braam@cs.cmu.edu.

Archive Index Issue Table of Contents

Advanced search

Magick with Images

Steve Whitehouse

Issue #50, June 1998

Mr. Whitehouse gives us an introduction to a free software package for manipulating images—ImageMagick.

Ever since the first computers with graphical displays, computer editing and transformation of images has been one of the most popular application areas. There should be little surprise that there is a multiplicity of free graphic software available under Linux.

Many of the available packages are targeted towards specific applications: there are many graphics libraries available for programmers which address reading and writing specific file formats. There are also image viewers available which allow interactive editing as well. ImageMagick is different in that it provides a comprehensive set of tools which can be used interactively in the X Window System or command-line driven. These tools are based around a common library, written in C.

For those who want to go beyond the capabilities of the precompiled programs (see the sidebar “Getting ImageMagick”), the library is extensible to allow the addition of different image file formats. The library can also be used from your own programs through its well documented API. In addition, both Perl and Python interfaces are available to allow image manipulation from scripts.

If you want to try out some of the features of the ImageMagick tools, many of them can be used on-line via the Imaging Machine at http://www.vrl.com/Imaging/.

It is impossible for an article such as this to cover every aspect of ImageMagick; therefore, I have written a short description of each of the major programs. A very good set of on-line documentation can be found at the ImageMagick web site—it covers all aspects of using and programming the system.

Display

Figure 1. Display of File Images

The display program loads images from a file and displays them on the screen (see Figure 1). It has a large number of command-line options which make it very flexible. Display is ideal for running from a script for a presentation or demonstration and will show video as well as still images. Images can be manipulated in a variety of ways, either from the command line or interactively. Figure 2 shows a set of images (the original is in the top left hand corner) that illustrate some of the effects available. In each case, the settings used are the default settings for that effect.

Figure 2. Display of Thumbnail Images

The display program is quite similar to John Bradley's **xv** image viewer. To display an image, you can either select one interactively from the menus (see Figure 3) or at the prompt type the command:

Figure 3. Image Magick Menu

```
$ display image.gif
```

In my research I work with many images in a raw format called YUV 4:1:1. This represents an image as a number of bytes. The YUV representation of an image is a combination of the luminance information in the image (a black and white version) called the Y component, and the chrominance information (the extra information required to make the black and white image into a colour image) called the U and V components. The Y component is thus made up of one unsigned byte per pixel in the image. Since the human eye is less sensitive to colour than luminance, the U and V components, each consisting of a number of signed bytes, are included at half the rate of the Y component. This gives a rather primitive form of lossy image compression compared to the more usual RGB representation where three unsigned bytes are used for every pixel in the image. In the examples here I will use a 176 by 144 pixel image. In order to display such an image, the following command can be used:

```
$ display -size 176x144\
image.yuv
```

In addition, if your image file has a header which is not among the supported types and you know that the image is in YUV format (this also applies to other raw formats), you can display it if you know the size of the header. For example, the following command:

```
$ display -size 176x144+16 image.yuv
```

displays a raw YUV image, with a 16-byte header.

Import

Import is an X image-capture program. It captures the contents of a target window, which may be specified in a variety of ways and stores it in a file. The captured image can then be viewed and manipulated with the following tools.

Animate, as its title suggests, displays an image sequence, which can be an MPEG file or a multi-image TIFF or MIFF file. The program works out the number of colours required to display the sequence before starting to play it so that the same palette can be used throughout to avoid interactions with colour schemes of the X Window System.

Montage combines several images into a single image. It is useful in preparing figures for papers and magazine articles, for example. There are options to annotate each image with text and to specify the background texture and colour and the border size. Figure 2 was created from the individual images by use of the montage program. The layout of the tiles and the labels were specified on the command line, and all other settings kept their default values.

Convert allows command-line conversion of images to and from many different formats. Its most common use is to convert to, and from, the ImageMagick MIFF format.

Mogrify provides a variety of different transforms which can be applied to images. A command-line interface to the transforms is available in the menus of the display program.

Identify prints all sorts of useful information about images, including a check on the completeness of the image and whether it is corrupt or not. Here is some example output:

```
$ identify image.gif
image.gif 106x80+0+0 PseudoClass 256c 13453b GIF 2s
```

In this case the image is in GIF format image, 106 by 80 pixels in size, has a PseudoClass colour map with 256 colours, is 13453 bytes in size and took 2 seconds to process. The **-verbose** option prints a more comprehensive list of information including the colour map.

Combine allows you to combine images in all manner of different ways. It can also be used to create difference images from two input images to see how they differ. There are a variety of ways in which images can be combined in addition to these two.

Programming with ImageMagick

While there are many packages available with some or all of the functions listed above, the real strength of ImageMagick lies in the ability to write programs using its library functions.

Included in the distribution is a simple program to demonstrate how to write your own image manipulation programs. It loads an image in JPEG format and creates a thumbnail in GIF format. I have changed the program slightly from its form in the distribution and presented it below. The thumbnail version will look something like the original image in Figure 1.

The C API to the ImageMagick library is documented through a set of web pages, which are also included in the distribution.

To compile the example code in Listing 1, you will need to give a command such as:

```
gcc -o example example.c -lMagick\
        -lX11 -lXext -ltiff -lpng\<\n>
        -I/usr/include/X11/magick -L/usr/X11/lib
```

The exact number of libraries required and the location of the libraries and include files will depend on the configuration of your system. The example given here works on my Red Hat 4.2 system installed from the RPM ImageMagick distribution.

To use the program, create a file called image.jpg and run the program in the same directory. The result will be a thumbnail-sized version of the original image called image.gif.

Using the included documentation, it is easy to see how this example can be extended and modified to form the basis of a wide variety of different functions. The same calls may also be made from Perl using the PerlMagick interface. Since I am not a Perl programmer, I have not investigated this interface.

Conclusions

ImageMagick is a complex package to use to its full potential; it is also very powerful. It offers a wealth of features in a flexible manner. It is easy to use the basic features without worrying about the more esoteric options available. I suspect that many people will use the basic options combined with only one or two of the more advanced options according to their application.

I consider ImageMagick a package well worth investigating for anyone needing anything from a basic image viewer to a full-fledged custom image manipulation system.

Pictures of Alan Cox are courtesy of Justin Mitchell and the ray-traced background image in Figure 1 was produced by David Beynon.

Image File Formats

Getting ImageMagick

Steve Whitehouse was first introduced to Linux, by the Computer Society at the University of Wales, Swansea, while studying for a degree in Electronic Engineering. Having gained his degree, he is now continuing his studies at Cambridge University by researching “Error Resilient Image Compression” and is sponsored by Racal Radio Ltd. If you want to contact him about the article or his research topic, his e-mail address is SteveW@ACM.org.

Archive Index Issue Table of Contents

Advanced search

Virtual Interview with Jeremy Allison and Andrew Tridgell

John Blair

Issue #50, June 1998

Author John Blair talks to two members of the Samba development team to discover some history and take a look at the future of the project.

Andrew Tridgell and Jeremy Allison together have written the majority of the code in Samba, the popular SMB (Windows Networking) file server for UNIX and UNIX-like operating systems. Andrew, the creator of Samba, is currently a Ph.D. student at the Australian National University. Jeremy currently works for Whistle Communications Corporation, located in Mountain View, California. Whistle provides Windows file sharing services from the InterJet Internet appliance using Samba. Part of Jeremy's job at Whistle is to develop and maintain Samba to make sure it works properly on the InterJet. Before conducting this interview I had “met” both Jeremy and Andrew through e-mail when both provided a valuable commentary on my book *Samba: Integrating UNIX and Windows*.

Jeremy Allison

Andrew Tridgell

This interview is dubbed *virtual* because it didn't actually take place as printed. I interviewed Jeremy over the telephone and his comments are taken nearly verbatim from the tape of that conversation, though some changes were made

in the interest of readability. Andrew's comments were added via e-mail after the fact.

John: The “history” file that accompanies Samba describes how Andrew wrote the precursor of Samba by reverse-engineering the DEC Pathworks protocol using packet dumps. Apparently he was motivated to do so because he had become used to using PC-NFS to access his home directory on a Sun machine, but couldn't run both the DEC Pathworks client and the PC-NFS client at the same time. How did you get involved with Samba, Jeremy?

Jeremy: At that time I was working for Vantive. I was porting Vantive products to NT on the quiet—they hadn't asked me to do it, but I knew it was something they would need. At the time there was no PC-NFS client for NT so I was actually doing the port using FTP. I would **ftp** the source files over, compile them, then ftp them back. This was a massive pain. Around this same time Andrew announced the SMB server 1.x—his second version. I downloaded it to take a look. It was Andrew's first networking program, I remember him saying, and it was a bit rough and ready.

Andrew: I didn't realize Jeremy looked at it so early. The first e-mails I got from Jeremy were around December 1993 at around version 1.5.20. That was after the nearly two-year gap when no work was done on the code by me or anyone else. Version 0.1 was released around December 1991.

Jeremy: I was actually porting a network service that worked exactly the same way that Samba does, so I started tightening up the networking code and adding some stuff. Andrew had implemented Samba just by sniffing the wire, because he's incredibly clever, where I can't do that sort of thing without a specification. Since I had the X-Open specification, I thought, why not extend this and implement Core Plus? So, I basically added the Core Plus protocol, which put in file locking and a whole bunch of other operations. That worked well enough for me.

Then Andrew leap-frogged me again and put in the LanMan 1 protocol. We kind of leap-frogged each other, but neither of us could figure out how to implement long file names, and we were stuck there for a while.

Then I was browsing in a book shop in Mountain View and I picked up a book on OS/2 programming, and it struck me that OS/2 already had to deal with this problem. Then I thought, this being Microsoft, what if they just linearized the long file name structures onto the wire? Well, worse things could happen, so I decided to try it. Luckily, I came down with the flu for a week, so I sat at home with my computer and it was one of those Zen programming moments where

you just code it up, try it and it works—never before or since! Later on we found a few bugs, but for the most part it worked.

Andrew: Yeah, the specification just said “level-specific structures” to describe the structures. We didn't have the hang of the trans2 stuff which was the real stumbling block. Once Jeremy coded up trans2.c, things really got moving as we had the infrastructure needed to do all the “modern” SMB features such as long file names. We worked a lot with the OS/2 header files for a while and those really helped. Jeremy contributed the trans2 code in October 1994 for version 1.8.0. This was a giant step forward.

John: So, how accurate is the official X-Open spec, from your experience?

Jeremy: The actual Core specification is pretty accurate up to the point where you get to the OS/2 support. As soon as you go into the OS/2 areas of the specification, it just disappears, and it's sort of like “here be dragons” on a map.

Andrew: Yeah, it just says “level-specific structures” and leaves it at that. You do need to remember that the main client (Windows) didn't have long file names when the X-Open specification was written, so not putting it in is not too unreasonable, particularly as a full description would be huge.

John: So the protocol is just whatever they could get to work?

Jeremy: Yeah, basically. The protocol is really horrible. It sort of grew like a wart. You can tell by looking at it. So, what happened was, once we got the long file name support working, the next thing we wanted to get working was encrypted password support. We hated having to put plaintext passwords out. There was a “magic constant” that Microsoft actually refused to give out to anybody. It's used in the DES encryption for the LanMan password support. They refused to give it to us without an NDA (Non-Disclosure Agreement). We replied “We're not giving you an NDA, because we're publishing source.”

There it sat for a while until I went to a Microsoft NT conference up in Seattle and had a conversation with a, uhh, sort of mentally challenged Microsoft person who was basically saying “Why do you want to break open all our LanMan system's?”, and I said “I don't, I want to interoperate with them,” which I suppose to her was the same thing. So, she put me in touch with a Microsoft programmer, Richard Ward, who is in charge of some of their security stuff. I chatted with him, and he couldn't tell me what it was, but he said, if you forward me a message I'll put you in touch with somebody who knows more about it.

Now what happened—and this is really interesting—the programmer he put me in touch with basically said, “Well, what do you need to know?” I thought, well, if I ask him the question directly, I won't get the answer. He'll say, “I can't tell you that.” So, I phrased the question differently. The LanMan encryption is a two stage operation, and we needed the middle bit. I phrased the question in such a way that we could work backwards from what he told us. I asked, if we encrypt this password, what will the second stage be? Once he told us, it was trivial to reverse it and out popped the magic constant. And once I had the constant, I actually took a look at a hexdump of the Microsoft redirector, and the magic constant is contained in a field of zeros as plain as day.

Andrew: It actually took us a day or two to realize that the information he sent gave us what we needed. Jeremy and I both realized it independently within a few hours of each other. It was an unexpected bonus!

John: So it was easy to see the magic constant once you knew where to look? [Note: the magic constant is **KGS!@#$%**.]

Jeremy: Yeah, once you knew where to look, but it did confirm that we knew exactly what it was. So, it took a while getting the encrypted password stuff. I did all the SMB password stuff, which I was a bit nervous about because it's a *setuid* program and I never do like writing setuid programs. After that I kind of left Samba alone for quite a while, ended up moving to Cygnus to work on Kerberos. This was when Volker (another Samba Team member and author of the SMB file system for Linux) came on board, and when I finally came back to it, Volker was asking “who is this guy?”, because I hadn't worked on Samba for a while.

We went to a CIFS (Common Internet File System) conference and Roger Binns, who wrote VisionFS (another SMB file server for some UNIX systems written by SCO), said to me “your share-mode support is completely broken, did you know that?” And I replied, “Is it?” Right after that conversation, Herb Lewis at SGI started doing some benchmarks, even going to Utah to benchmark Samba against other programs such as the Syntax (VisionFS) code. Herb made the remark, “I can't get NetBench to run because I keep getting share-mode violations.” I remembered what Roger Binns had told me so I started looking at the share-mode code. The code was a tremendous piece of work, but it wasn't entirely safe—it only stored one share mode for each open file. We now have a completely correct share-mode implementation. So, in other words, we can have multiple share-mode entries for each open file. Once I put all of that code in—this is around 1.9.17, we could run NetBench. We got reasonable results—they were okay—probably as good as Syntax's.

At this time I realized that what we needed for performance was *oplocks* (“opportunistic locks”). About at the same time Herb, who's actually had a great effect on at least my parts of Samba code, started complaining that his browsing was broken—he couldn't make his browse lists be seen across subnets. Luke (Luke Kenneth Casson Leighton—another Samba Team member) had written all of that stuff and I had always thought it worked. Basically, I went and played on SGI's network, and I realized that there were things about the browsing that were broken. So that's when I started being involved with the browser stuff—I started fixing up Luke's code.

There have now been three implementations of the browser code. The first was a single-threaded completely blocking one that Andrew did which basically got it up and running. Then Luke took that and turned it into an event-driven model that was truly an amazing piece of work, very well done. But he was still learning the protocol at the same time. So, I hacked up some of that code section, and it sort of worked, but I realized that in order to make it completely robust, I had to keep the same event model and data structures but completely rewrite the code. So that was what I ended up doing. That effort was actually spurred by the fact that Whistle needed browsing to work correctly over semi-reliable links, i.e., links that would go down periodically. I began by not rewriting it—just fixing that particular bit, but I ended up completely rewriting it. It was one of those things where, you know, after a while you see you've completely rewritten something without meaning to.

Andrew: Actually the first version wasn't completely blocking—it used recursion to simulate some properties of threads (a *really* ugly idea). It worked but could easily run out of stack. Luke's implementation fixed that problem and also added a lot of new code.

Jeremy: So, then, I knew the oplock implementation was possible. I had a design for it, but something like six months had passed before I actually had time to sit down and do it. The oplock code is probably the most complicated part of Samba right now.

John: It's probably the most complicated part of Windows networking.

Jeremy: Well, yeah. That took a long time to get right. Typical with SMB, there were a lot of ugly implementation details that aren't in the specification. For example, when someone does an open that needs to break an oplock, do you check deny modes first? Now, logic would dictate that you do, because if the deny modes disallow you from opening the file then there's no need to check the oplock. It turns out if you do that everything breaks. My first implementation did that as an optimization because I thought, well, this will be really fast. It turns out if you do that a bunch of programs break because a lot

of the Windows 95 redirector internals expect that a file could be opened twice and the second open would cause the oplock to break even though it's still going to get a share-mode violation.

John: Do you suppose the people that wrote that were optimizing for their own server code?

Jeremy: I don't know. Previously they had only tested against their own server. Probably there are a really weird bunch of hacks in there that work against their own server and that we have to emulate. Of course, whenever the redirector is broken and it doesn't work with Samba, people say that Samba is broken.

John: To change the conversation a little bit, are you paid by Whistle just to work on Samba or do you have other responsibilities?

Jeremy: I have other responsibilities, although getting Samba working has been a big part of my job. One of the issues right now is that Samba works well enough for Whistle, so I might actually have to work on some other programs. However, I really do love to work on extending Samba. My current plans for 1.9.19 are to do NT SMB, which will implement *change notify* and many other options. One of the things Whistle did that was actually very good is they drove me into doing dynamic internationalization support for Samba.

John: Any final comments?

Jeremy: One more thing that you should include—my wife is the product marketing manager for Network Appliance, a company that makes an SMB implementation. So, my wife and I are in direct competition along with Samba and Network Appliance, which is really amusing. She gives sales training which includes a competitive analysis of Samba. She says people always ask, “What kind of freaks write this stuff and then give it away?”

John: That reminds me of conversations where I'm trying to explain the GPL. It took me a while to fully internalize it and understand its implications, and explaining it is a long conversation—sort of like explaining a political theory.

Jeremy: Yes, it's a revolutionary idea. Like most revolutionary ideas it will end up changing the world, and it takes a while to get. I won't release software under any other license now. I did before, and I've always considered it a mistake and regretted making it. The **pwdump** stuff I released under the BSD license, and I wish I hadn't now. I see that code reused by other people, but because of the license I don't get to see how they're using it—they don't have to give back.

John: Andrew, why did you choose to release Samba under the GPL?

Andrew: I didn't at first! Server-0.1 and all versions prior to 1.5 were under a “if you want to use this for commercial purposes then contact me” license.

Version 1.5 came out after the 2-year break and after I had started using Linux. I was impressed with Linux and chose the GPL not because I understood it (I didn't) but because it was what Linux used. Now that I understand the GPL I'm very glad I chose it. It encourages just the sort of development effort that is needed for Samba.

John Blair is currently a system administrator in the University Computer Center at the University of Alabama at Birmingham where he tends several UNIX and (shock, horror) Windows NT servers. By the time this article is published he may be working someplace else. John is also the author of SAMBA: Integrating UNIX and Windows, published by the same people who bring you *Linux Journal*. Feel free to contact him via e-mail at jdblair@uab.edu.

Archive Index Issue Table of Contents

Advanced search

Linux WAN Routers

Tony Mancill

Issue #50, June 1998

Here's another great use for Linux; Mr. Mancill tells us why his company picked Linux routers over the big names.

Every time I deploy a Linux system for my company, the phrase “Linux in a production environment? It'll never happen,” stills echoes through my head. At a previous employer, this was the pat answer to all of my queries as to when we could try Linux out in our product evaluation lab. I have since had the chance to use Linux to solve real-life problems, and I am ready to report—“It happens every day!”

This article will discuss the advantages of Linux-based WAN routers in terms of total cost of ownership. Although many technicians may find this approach unsavory (it certainly does not appeal to the idealist in me), the truth is that most finance departments are rarely interested in the technical elegance or excellence of their IT departments. In the eyes of those signing the checks, value is more important. Cost includes not just hardware and software, but all related personnel and maintenance costs.

In today's penny-conscious corporate environment, technicians need to be cognizant of the fact many companies offer routing as a service, which may (at least on paper) look less expensive than your salary and equipment. Therefore, it makes good sense for network administrators to remain conscious of the value they are providing their employer. If you are a solutions-provider, this article may help you increase your profit. And for those operating with a limited budget, deploying Linux routers may be the only choice for connecting sites to each other or to the Internet.

Costs aside, in my opinion Linux routers do possess technical elegance and excellence. I will focus on the functional “niceties” of this platform—plus some day-to-day experiences. For those of you already familiar with Linux, it might be interesting to see how it is being used in a 24x7 production environment. For

those not yet using Linux, this article will acquaint you with some of the possible applications of this versatile and stable platform.

Figure 1. WANPIPE S508/FT1 Card

From this point on, the term “Linux router” will be used to refer to an x86-based PC running Debian/GNU Linux and outfitted with Sangoma's WANPIPE S508 router card (Figure 1). After using this platform as an alternative to “BigName” traditional routers for more than 18 months for frame-relay and Internet routers, I am a strong proponent of this solution.

Cost

Linux routers are economical both in terms of hard costs and the associated hidden costs in providing a routing infrastructure. These costs include:

- Telco access (+ usage-based charges where applicable)
- Router hardware
- Router software, upgrades and support
- Personnel costs, including salary, training and maintaining the router during day-to-day operations for both troubleshooting and upgrades
- Lost productivity and revenue due to downtime—in the holistic view of your company's management, often quite expensive

For usage-based access methods (e.g., most types of ISDN), monthly costs depend upon the connect-time. In this case, it is beneficial to control when and for what reason a connection is initiated. Many routers cannot provide this sort of control at all; a Linux router comes equipped with schedulers and scripting languages.

Router hardware costs can vary wildly, depending upon the interface types and speeds, protocols supported, capabilities provided (such as packet-filtering) and switching speed. For less than the cost of the least expensive traditional router that supports a V.35 interface, you can have equivalent connectivity with superior functionality and supportability using a Linux router.

An easily overlooked cost of working with digital circuits (other than ISDN) is the CSU/DSU, which is used to interface your router to your telco access. This device understands the signaling on the digital access line, e.g., a T-1, and converts it into a bit stream on a V.35 interface. They can be expensive. The Sangoma S508 is offered with an integrated CSU/DSU which saves money and makes cabling and mounting easier.

The cost of router software, hardware and software upgrades, as well as the cost of yearly support agreements for router hardware and software, can be significant. (Traditional support is often 10 to 15% or more of the new price of the hardware per year.) These costs approach zero for the Linux router solution. The operating system, including tools and upgrades, is free. The PC hardware is inexpensive, and because the requirements are so modest it can often be inherited from others trying to upgrade their desktops for more horsepower. (All of my systems are “hand-me-downs”.)

Even more important than the base hardware costs, Linux routers offer investment protection since you have a clear upgrade path for all aspects of your router. Additional links can be added for cost of another Sangoma card. Mixing links and media types is simple and inexpensive. For example, if you wish to upgrade your LAN backbone to 100Mbps or to ATM, adapters for our BigName router cost about $4000.00 each, but for Linux any decent 100Mbps Ethernet card will work fine. Faster switching is merely a motherboard/CPU upgrade.

Now and then someone will quip that a PC is not fast enough to be a WAN router. I think this statement shows a lack of imagination. If this were true, there would be no point in having 100Mbps Ethernet cards. How can you expect your desktop to send or receive packets at 100Mbps when it is not able to read+send packets arriving at 1/100th of that rate?

Packet-filtering, address translation (IP-masquerading) and proxying are often add-ons for traditional routers. By contrast, adding this functionality to a Linux router is free, and easier to install and manage.

BigName router software upgrades can be time-consuming. Unless you have spare BigName routers, practicing your upgrade is not an option. This is even worse when you have both “BigName X” and “BigName Y” routers. Different procedures, different problems and phone support for any of them can be expensive. Having more than one closed-system router vendor also means more money for training and more “fragmentation” of the skill-sets of your support staff. Instead of three capable generalists, you have one person trained for X, one for Y and a third who tries to keep up.

This leads to another part of the cost of providing routing services. How much does it cost to have people tend to the environment? Salary, training, time spent configuring, developing reports, upgrading and troubleshooting are all part of the total cost of ownership. I would say that this is the most important reason to seriously consider Linux as a router platform. First of all, there is an ever-increasing supply of talent that has experience with Linux. This keeps salaries for support staff reasonable. (Try to find enough money to hire a BigName specialist.) Even salty UNIX administrators feel at home on Linux systems, once again increasing the resource pool, providing backup support and easing cross-training within your IT department. This element of commonality cannot be stressed enough. Secondly, configuring a Linux router uses the same tools as configuring the network card on any UNIX system. Anyone who feels comfortable at a shell prompt and understands TCP/IP is a potential resource. This is important, because at some point, your environment will need support.

Maintenance of WAN networks tends to be infrequent but intense. Problems tend to occur at 2:00AM six months after you last touched your BigName router. (You are most likely at home, sitting in your bathrobe, dialed-in to your office. What are the chances you have the manuals at hand?) By contrast, a Linux router uses many of the same tools you work with every day, and you have all of the documentation on-line in the form of man pages or text files. You may not have worked with the router for a while, but you use **ifconfig** and look at /var/log/messages each day. Even the hardware-specific tools tend to be more fully featured and easier to use. For instance, Figure 2 is a screen showing Linux PPP statistics as monitored from an attached workstation.

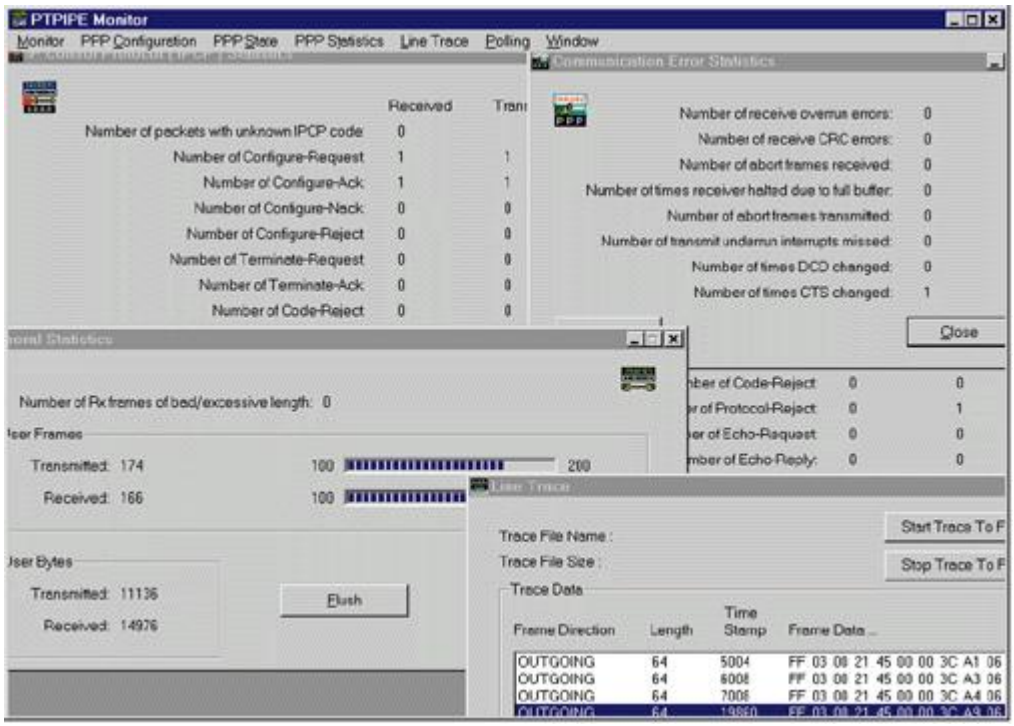


Figure 2. Screenshot of Linux PPP Monitoring on Attached Workstation

For day-to-day troubleshooting, you have a whole suite of tools at your fingertips that you can use to hack your way around the problem. And because these tools are familiar, your problem resolution time is shorter. A *keep-alive* ping script may not be the prettiest solution, but it will keep you out of reactive mode long enough to research the real problem. When I was younger and very naive, I believed that when you bought a piece of hardware or software from a BigName it was fully debugged. This is ridiculous—all code has bugs in it. The real question is—what options do you have to deal with these bugs?

Linux in the Production Environment

Another plus for Linux routers is they can run additional services and programs. At your disposal is an entire OS family of services and applications with 20+ years of debugging and fine-tuning. Because UNIX has always promoted a tool box philosophy, the only real limit to what other services your routers can provide is your imagination. Here are some ideas:

- Deploy secondary services for more redundancy. For example, a router can act as a secondary DNS server or a secondary SMTP gateway.
- Configure the router for a remote site to act as a slave DNS and a caching HTTP proxy—perhaps even Samba for file and print services.
- Deploy a "one-stop-shop" Internet router—use the same machine for the external DNS name server, an SMTP mail-exchanger, an NTP server for your site and an Internet firewall.

Linux also provides flexible packet-filtering, SOCKS, proxying solutions, PPTP and IP-masquerading. For those who have special needs, there are modules for shaping traffic patterns (throttling the bandwidth of certain types of traffic or traffic between given hosts).

Deploying multipurpose systems also makes sense in terms of reliability. Having fewer concentrated points of failure typically increases the average uptime of an office, because the possibility of hardware failure is equal to the sum of the possibilities of failure of the individual components. Therefore, fewer multipurpose systems should result in a lower overall chance of failure. Be careful though: lumping services together means that multiple services will be down simultaneously. You should always consider the interaction of the services you are combining and try to combine services that would be useless anyway if the router were to fail.

It is hard to exaggerate the importance of reliability for a routing platform. Unreliable systems do not just cost money: they can utterly kill your business. Worse than that, they can cause you to get paged during non-work-related activities (like sleeping).

Tips for Running Linux in the Production Environment

You might find some of the following tips helpful. Many of them were learned the hard way.

- Back up your system regularly. (I have had good luck with BRU from EST Software.)
- /var should be a separate partition because it is going to hold all of your spools and log files.
- Make a set of rescue floppies. My rescue floppies include Debian rescue disk, bootable DOS disk with the configuration utilities for my Ethernet cards and bootable DOS disk with Sangoma's snooper.exe (a sniffer utility for their communications cards).
- Keep all of your system scripts in a common location, e.g., /us/local/bin or ~sysacct/bin and use a common header format for all of your scripts that includes a history of changes. This saves you time by reminding you of what you did and when, and helps others who may inherit or need to modify your scripts. The first 2-letter field contains initials, when more than one person has access to your environment. All my scripts start like this:

```
#!/usr/bin/ksh
# name_of_script [cmdline args]
# [description of script, if needed]
# modification history:
# tm970612initial release
# ls970923added functionality X
# tm980115most recent edit
MAIL_TO="tmancill@us.lhsgroup.com"
```

- Remain security conscious. Change passwords regularly, and use **ssh** or a similar program to protect against internal TCP/IP snoopers. Use PGP or a similar program for sharing sensitive information with others.
- Have spare components available. Linux is tolerant of changing hardware, so you can move the hard drive to a different system without problems. A spare communications adapter and an IDE hard disk loaded with the base OS plus kernel source should suffice.
- Reboot your system after any (major) configuration changes, e.g., changes to the routing tables specified in the start-up scripts. If this cannot be done right away, schedule a time to do it. This is suggested not because Linux needs to be rebooted for the changes to take effect, but more because it runs so long without needing to be rebooted. The “reboot-test” provides some insurance against Murphy's Law. Otherwise, your system will most likely be rebooted when you are not around, by an NT-administrator trying to log in with **CTRL-ALT-DEL** six months after you have made any changes to it whatsoever. If it cannot restart without intervention, the NT-administrator will be mucking about on your system

until you show up and are confused as to why the system is acting funny because you cannot remember the last changes you made.

- *Understand* TCP/IP—ports, routing, variable subnetting, DNS, the differences between TCP and UDP, etc. This cannot be overstated. Downtime statistics for large production environments are alarming in that they often show human-error as the number one cause of failure. (Linux test beds are cheap; practice on a spare system. Because communication adapters under Linux work just like Ethernet devices, you can simulate your WAN environment with extra Ethernet cards and a separate Ethernet segment.)
- Keep documentation on your systems. It does not have to be much—just note how each system varies from the base OS load.
- Keep your head when things get hectic.

Additional Benefits of Open Systems

People sometimes tell me that GPL software, because it's free, does not have the quality to be part of a production environment. This is like saying that only authors with publishing contracts write good poetry. When I hear this, I always have to wonder if these people have ever even used GPL software or know how much they depend upon it every time they browse the Web or receive e-mail from the Internet.

There are some very distinct advantages in source code availability for network administrators. As the Internet continues to evolve, along with protocols, security measures and resource conservation techniques, routers will have to keep up.

Five years ago, the shortcomings of IPv4, and the need for protocol encryption and encapsulation might have been far-fetched ideas, suited only for the minds of IETF gurus. Today you deal with them each time you use IP-masquerading or PPTP. Because of its openness and its rich tool set, Linux makes an ideal platform for developing and testing these sorts of protocol extensions (e.g., IPv6 and ENSKIP). As a network administrator running on Linux, these tools will often be available to you sooner than in commercial implementations. And they will be written by someone who not only wants to see the software work, but also uses the software himself; not by someone trying to meet a coding deadline. Because Linux is not in commercial competition, the focus is on interoperability, not on proprietary protocol extensions. Looking forward, it is difficult to say what we will be running in the year 2005. I do, however, feel certain that developing these tools in a robust, open operating system must be substantially easier than developing for proprietary architectures with more limited tools and support. Therefore, I feel better supported.

Good support and vendor stability are essential aspects of any large IT investment. I never have to worry that Linux is going to go out of business or be purchased by another company and then discontinued. As for WAN routing hardware for Linux systems, I have the feeling that it will be around as long as there is a market for it. If my communications hardware vendor does ever go out of business, I'm not left hung out to dry as I would be with a traditional router. I have the source code for the drivers and can continue to adapt and enhance for as long as it is functional and economically feasible to do so.

Just because you paid for support does not mean that you will get it. Arguments to the effect that “we cannot use Linux because we cannot get support” are flawed. Typically, they are made by a management that does not believe employees are capable of performing their jobs. The largest percentage of my experience with vendor support can be categorized in one of these ways:

1. Completely wasted time trying to explain the problem to someone who has no idea what I'm talking about.
2. “Have you tried our latest patch/reloading the software?”
3. “It sounds like it has nothing to do with our system/software.”

Troubleshooting IT problems can be difficult and time-consuming, and no one can afford to staff their help desk with their top programmers and troubleshooters. So since you will have to troubleshoot a large majority of your problems yourself, why pay for support?

Linux as a Router in Production

My firm has offices in the United States, Europe and Asia. As an international company, WAN connections are an important part of the infrastructure. Because of the time zone differences between our sites, it is critical to have a stable routing platform; midnight in one location is high noon in another, so maintenance windows are small. Just like any other company, we are conscious of costs. To meet these goals, we use Linux/Sangoma routers for:

- 512Kbps link to the Internet
- 1.5Mbps frame-relay
- 56Kbps backup link to the Internet

We intend to deploy three more Linux/Sangoma frame-relay routers this year. In addition, we use Linux as a LAN router, a server platform for all of the standard TCP/IP services (DNS, FTP, HTTP, packet-filtering, IP-masquerading, proxying, SMTP, NTP, NNTP, etc.) and, of course, as a desktop.

The actual configuration of our Linux frame-relay router is GNU/Debian Linux (version 1.2) running on a 486/66 with 8MB RAM, a 850MB IDE hard drive, a Sangoma WANPIPE S508 router card and a SpellCaster DataCommute ISDN card. The ISDN card is used as a backup, in case the frame-relay fails. This system had been up for over 160 days before it was rebooted by a sadly mistaken NT-administrator trying to log into another system that shares the same keyboard and monitor.

If you're wondering why I went to the trouble to write an article about using Linux as a router, maybe the following anecdote will help explain it.

Once upon a time, our Internet link was connected with a BigName router. One day, this router decided to die. In total, it took about an hour to get a technician from BigName on the phone; we whiled away the time scrambling around looking for our support ID, wading through the “press six if you'd like to use our fax-back server” menus, waiting on hold and fending off frantic users. After a short discussion about my abilities to configure a terminal program (peppered with a few curt remarks of my own about what sort of idiot cable was needed to access the console), the technician decided that we needed a new motherboard. Since we had paid copiously for our support contract, a new board was to arrive the next day. We informed our users of the situation and eagerly awaited our package. A package did arrive promptly the next day by the promised time. However, much to our dismay, we had received a new power-supply and case—no motherboard.

Now we were in trouble. BigName was going to send us our part, but that meant at least another 24 hours of downtime. Based on our experience with the Linux frame-relay router, we decided to try our spare Sangoma S508 card for this link. We had Linux loaded and the software configured in about an hour. We started the WANPIPE software and nothing happened. Using the **ppipemon** utility that comes with the Sangoma product, we were able to tell that the link was failing in the LCP negotiation phase. That is, our router was talking to the ISP's router, but they could not mutually agree on an operating parameter set for the link. It is fortunate that we had these tools. Our ISP was telling us that they were quite certain that we had no routing hardware whatsoever attached to the line. This despite the fact that we could tell them the exact data streams we were receiving from their router.

In desperation, we called Sangoma to see if they were familiar with this sort of behavior. They were not, but offered to look at the output of a data trace. We collected a few seconds of the failing negotiation sequence and mailed this to Sangoma. Less than four hours later, I received a call from an engineer at Sangoma who told me there was a nebulous portion of the PPP RFC which had been implemented by our ISP's port multiplexor. Best of all, Sangoma had

already placed a patch on their FTP server. Fifteen minutes later we were up and running. Although the motherboard did arrive from BigName, we have never gone back. This router sits in storage as a backup to our backup. In looking back at the sequence of events, I am impressed by the following:

- We were better equipped with tools to troubleshoot problems than our ISP. Maybe we were just more motivated, but I have to question the integrity of either the technician or the tools when I am interrupted while listing the sequence of LCP packets with "Are you sure the router is powered on and attached to the CSU/DSU?"
- We were able to get a patch in less than a day.
- We were able to turn an outage of at least 48 hours into less than 30, and it would have been even less than that if we had been quicker to consider using the Linux router. (In a production environment that strives to have 99.5% availability, you have 43.8 hours a year for maintenance and downtime.)

Conclusion

The intent of this article is not to say that Linux routers will make traditional routing hardware obsolete. When considering routing hardware, make sure that the tool fits the job at hand. If you have a T-3 to the Internet or want to tie together remote sites with ATM, you probably need to be shopping for equipment designed explicitly to switch and route packets at those speeds. By the same token, why go to the extra expense and trouble to deploy a special-purpose piece of hardware, along with all of the inconveniences that come with it, when you only need to route 128Kbps or even 1.5Mbps?

Because no one can foresee all of the demands that will be placed on their routing environment, flexibility and expandability are desirable in any solution. The Linux kernel is rapidly supporting increasingly more sophisticated types of traffic-shaping and packet monitoring. Routing hardware, including the processor, can be upgraded inexpensively. Furthermore, this same hardware can provide additional functions. Finally, a Linux router comes equipped with a complete set of familiar tools for monitoring and customization.

For a minimal investment in hardware and time, you can try a Linux router for a new link or to act as a backup for your current link(s). If you are new to data communications or need support, you are more likely to find a Linux hacker who can read (which is all it takes to get a Sangoma card running) than to find a BigName router guru. Typically, Linux folks are pretty friendly and willing to help. After all, some of this stuff is just neat. For business environments, the availability of Linux talent is increasing, and training for this environment is substantially less expensive than for closed-systems. Because Linux is open,

your investment of time and capital is better protected. Give it a try. You will not regret it!

Glossary

Tony Mancill spent a lot of time studying Electrical Engineering before graduating from Georgia Tech, only to end up playing with computers all day. When he's not working at LHS Communications Systems, he divides his time between playing the drums, home-brewing his own beer and trying to teach his dog new tricks. He has recently volunteered for the GNU/Debian Linux project as the maintainer of the **wanpipe** package. You may contact him via e-mail at tmancill@lhsgroup.com.

Archive Index Issue Table of Contents

Advanced search

Linus Speaks at SVLUG Meeting

Chris DiBona

Issue #50, June 1998

Linus Torvalds tells the Silicon Valley users group about his current work on the Linux kernel.

When we gave the job of arranging speakers to Sam Ockman, we never doubted his ability to bring in terrific speakers. His first speaker, for our January meeting, was his personal hero, H. Peter Anvin. For February, we had two speakers: Eric Raymond of “The Cathedral and the Bazaar” fame, and Bruce Perens of Debian. Therefore, the question became “How do you top these Linux luminaries?” Sam's answer was our March speaker, Linus Torvalds.

Until December, the Silicon Valley Linux Users Group (of which I am the Vice President) met in the back dining room of the local Carl's Jr. (local burger chain). Carl's could hold around 40-50 people tightly. We had been talking about moving the meeting from this spot for some time, and some of our members who worked at Cisco pleaded to get a room for us for the meeting in February. As expected, this meeting was standing room only, and we knew that with Linus coming we needed a much larger space. Again, Cisco (with Ben Woodard pushing it through) came through for us with a room spacious enough for 350 people in their Gateway Conference center.

Figure 1

Figure 2

About half an hour before the meeting began, the chairs were full and people began to sit on the floor, against the walls and all around the room. Approximately 500 people had come to hear Linus speak. We were lucky—the air conditioning was in good shape, and the fire marshal didn't show up.

The meeting began and after the user-group formalities were complete, Linus was presented with the VA Research Excellence in Open Source Software

Award, co-sponsored by Red Hat and O'Reilly. The prize was a loaded dual Pentium 333 from VA Research. In fact, Linus made out very well, receiving not only the computer just mentioned, but also a PalmPilot professional from Klaus Schulz of 3Com and a six-pack of real beer from local legend Rick Moen. It should be noted that Linus didn't know about any of these awards before coming to the meeting.

Figure 3

Accompanied by thunderous applause, Linus stood before the podium and expressed his shock at the number of people who had shown up to hear him speak. He had been under the impression that it would be a small, intimate meeting like the first one he attended last year at the burger joint.

He began his speech by telling the group what he wouldn't be talking about. He said he wouldn't be talking about user issues or MIS issues—all he would talk about was what he was doing with the kernel.

Figure 4

The hour-long speech (not counting the Q&A afterward) was a technical discussion of how he is improving SMP (symmetric multi-processing) support in the kernel. He talked about the challenges of moving from a single kernel resource lock for all CPUs as in the 2.0.x kernels to individual resource locks for the new kernels. He discussed the ways these changes affect things internally for the kernel, and how it affects the handling of shared memory, interrupts and I/O. Linus also spent some time talking about how the file system is being changed internally for better performance. The speech will be available on-line by the time this article goes to print (see Resources); I'd recommend that you download it and listen to the whole thing.

Figure 5

In addition to videotaping the speech and taking still photos, the meeting in its entirety was broadcast over the MBone (Internet protocol multicast backbone). After Linus had finished speaking, the door prizes were given out, and everyone left the meeting happier and smarter than they were the day before. Special thanks are due to Linus Torvalds for speaking and to everyone else involved in making this meeting a success.

Resources

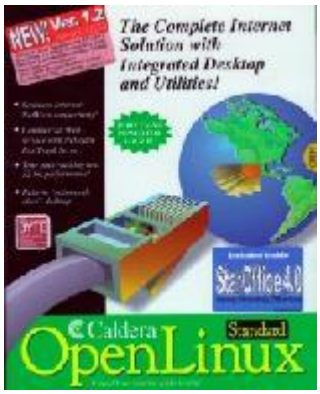
Chris DiBona is a computer security specialist for StrongCrypto Inc. He can be reached at chris@dibona.com. His personal web site is located at http://www.dibona.com/.

Caldera OpenLinux Version 1.2

Sid Wentworth

Issue #50, June 1998

Caldera continues to offer a commercial grade, shrink-wrap package with more of the applications to make it a complete solution for business.



- Manufacturer: Caldera
- E-mail: orders@caldera.com
- URL: http://www.caldera.com/
- Price: $59 US (Base) $199 US (Standard)
- Reviewer: Sid Wentworth

Caldera's latest offerings are OpenLinux Base and Standard, version 1.2. Both are based on the Linux 2.0.33 kernel which, at the time of this writing (March), puts them a jump ahead of the other commercial Linux distributions.

However, a new kernel is not what makes this distribution stand out from the others. Caldera continues to offer a commercial grade, shrink-wrap package with more of the applications to make it a complete solution for business. I don't have the numbers, but I would expect that the majority of Caldera products are sold through VARs (value-added resellers) who offer complete solutions to their customers.

Specifically, applications found in COL, in addition to the programs generally found in all the distributions, are:

- The Netscape FastTrack secure web server
- The Looking Glass Desktop
- Netscape Communicator 4.0 and Navigator 3.0 web browsers
- StarOffice 4.0
- Adabas D personal edition SQL database
- BRU 2000 backup/restore utility

COL Standard includes commercial versions of these software packages. COL Base is missing the Netscape web server and BRU, and it includes only the personal-use versions of StarOffice and Adabas. Frankly, I consider the missing pieces to be worth the price of the standard distribution.

By the way, Caldera is the company that established the relationships to make the ports of Netscape, StarOffice, Adabas and WordPerfect to Linux happen.

Installation

Caldera has their own install/systems administration tool called LISA. I find it to be very much like S.u.S.E.'s YAST. I consider this interesting as both were written in Germany—it must be an indication of German engineering.

LISA is easy to use with dialog boxes to guide you through the installation. The installation instructions come in English, German, French, Italian, Spanish and Portuguese. LISA offers you a choice of a few standard installations (small disk, reasonable-sized disk, large disk and go for it), plus, for the expert who wants to do everything himself, an option to fine tune the packages being installed.

I did the install on an AMD K6 system with a huge disk. I first selected the “go for it” install option and ran into some glitches—nothing serious, just messages like “couldn't find unknown.rpm”. For round two I selected the default installation and everything ran flawlessly. Hopefully, by the time you read this, the kinks will be worked out of the everything install.

The only other problem I encountered during installation was an error in configuring the Korn shell. The location where the installation program expected to find it (/usr/bin/pdksh) was not where it actually was. A quick edit of the /etc/passwd file and all was well.

The User Environment

Caldera, by default, uses the Looking Glass Desktop. Not being a desktop sort of guy, I am not particularly excited about it, but, if you want a desktop, it seems adequate. My biggest complaint is that this desktop is non-standard; however, this is similar to my complaint about Red Hat's use of a totally non-standard X Window System manager setup called “The Next Level”. Just give me an xterm and I am happy.

The biggest hit for me was StarOffice. As I said, this product alone is worth the cost of COL standard. If offers a word processor, presentation software, graphics software, spreadsheet and HTML editor all in one integrated package. My biggest objection to StarOffice is that it looks like a Windows 95 application, but I expect this was a conscious design decision and probably the right one for most potential users. I understand *Linux Journal* will be doing a review of this product in the near future.

I didn't get a chance to play with the Netscape FastTrack Server but this is a commercial, secure web server so I expect it will be very useful to anyone intending to set up a serious web site. A separate, 130-page manual is included in the package that covers administration of this server.

The Package

Caldera OpenLinux comes in a very nice package of serious store shelf display quality. Inside the front cover of the box is all the appropriate marketing stuff with quotes from *Unix Review*, *Software Development*, *InfoWorld*, *Computer Shopper* and *Linux Journal* about the product. Current Linux users don't need this hype, but it certainly helps on the retail shelf.

There is a bootable CD-ROM with the base distribution on it and a boot and modules floppy for systems that can't boot off of CD-ROM. There is also a second CD-ROM labeled “Extra” which includes StarOffice 4.0 and some programs contributed by outside sources. A third CD-ROM contains Adabas. There are also cards for obtaining a discount on Accelerated X servers, PartitionMagic and a free issue of *Linux Journal*.

The manuals include a *Getting Started Guide*, an addendum to that guide and the *FastTrack Server Administrator's Guide*. The *Getting Started Guide* does a good job of hand-holding for the installation process. Most of this is unnecessary if you have installed any Linux flavor before, but it is there if you need it. What is important is the information on using the Looking Glass Desktop.

The addendum is a release summary for version 1.2 and some additional information not covered in the manual itself. My biggest nit is that StarOffice 4.0 is mentioned on page 7 of this manual in the form of a description, but there are no installation instructions for it. Installation isn't hard, but it would have been nice to cover it in the documentation.

Conclusion

COL Base and Standard are real products with a lot of retail appeal. They aren't perfect, which disappoints me—the problems I found could have been avoided by a little more testing before burning the CD-ROMs—but the competition (including Microsoft software) isn't perfect either. The included applications are powerful and show that Linux on an office desktop is truly possible.

For the average geek, Caldera's products are not much different from Red Hat or S.u.S.E. For serious office applications or convincing your significant other to use Linux, I feel OpenLinux has a lot of potential.

Sid Wentworth currently resides in a small coastal town in western Washington where he writes anti-spam code, forages for gui-ducks and brews virtual beer. He can be reached via e-mail at info@linuxjournal.com.

Archive Index Issue Table of Contents

Advanced search

Red Hat Linux 5.0

Michael Taht

Retro

Issue #50, June 1998

This review details my experiences with 5.0 in the two months since its release. I maintain 17 Linux machines, 5 of which are now running 5.0.

- Manufacturer: Red Hat Software, Inc.
- E-mail: info@redhat.com
- URL: http://www.redhat.com/
- Price: $49.95 US
- Platforms Reviewed: Intel and Alpha
- Reviewers: Michael Taht and Retro

Red Hat 4.2—with its associated 30+ patches and upgrades—has been my standard operating system for all my clients since its release. Its ease of use, reliability, simplified install, corporate support and TheNextLevel X Window System interface have made it my preferred Linux. However, I was beginning to feel that 4.2 was getting exceptionally long in the tooth, over 6 months old—two Internet *years*. Therefore, I was overjoyed when 5.0 was released on December 1, 1997 and delighted when Red Hat started to ship Netscape 4.04 as standard in late January.

This review details my experiences with 5.0 in the two months since its release. I maintain 17 Linux machines, 5 of which are now running 5.0.

Red Hat 5.0 can be installed via CD-ROM, hard disk, NFS or FTP. Red Hat 4.2's SMB install option has been dropped. It's a simple matter to drop an FTP server onto a Windows 95 or NT box, so I can live without SMB support for installation.

I installed the commercial release of the Intel version from CD, and Retro installed the freeware release of the DEC Alpha version via FTP. Most of my computers support booting from CD-ROM, but the CD-ROM only booted on two of my most recently built machines.

The NFS and FTP installation options are best used on a file server on your internal network. Installing via FTP over a T1 line or less is painfully slow, so you are better off pulling down a mirror of the release overnight and installing from a local machine. If you have multiple machines to install or upgrade, NFS and FTP are convenient, allowing you to update or install new versions of Red Hat on several machines at the same time. (You should copy the distribution to the hard disk, as CD-ROMs don't handle concurrent access very well.)

Performing the base upgrade is simple, but getting everything working again is not so easy. The CD-ROM release with prebuilt installation floppies arrived in my mailbox. I inserted disk one into my first computer, but it wasn't needed—Red Hat booted from the CD-ROM. I selected automatic install, then upgrade. Red Hat then prompted me to override or add additional packages. Because I was unfamiliar with the packages already installed in this machine I decided to just take the defaults and go with the flow. I regretted this later. The remainder of the automated upgrade procedure took about 15 minutes on a 24x CD-ROM, then prompted me for X Window System and network information.

Rebooting was slow and fraught with errors. When the machine finally came up, it turned out the network interface was incorrectly recognized. No problem, I said, I'll just recompile the kernel with the correct options.

```
make xconfig; make clean; make dep; make zlilo
make modules; make modules_install
shutdown -r now
```

Here's where I have a beef with Red Hat—their penchant for “pristine sources” means that by default both /etc/lilo.conf and the /usr/src/linux/Makefile need to be modified for a new kernel to boot correctly—a very confusing thing for first timers. I recompiled and rebooted again: **sendmail** hung, AMD broke, **httpd** broke, Samba hung, Povray broke. The list of broken programs was quite long, and although fixing each problem took only a few minutes, it was days before I had a completely functioning system again. Now that the installation process is

over, the boxes I have are very usable and stable, but I really don't want to have to go through that again for all my production machines.

Red Hat 5.0 features an automated scripting utility that allows you to upgrade or install on a number of machines. Unless I had a large number of very similar machines (like the rendering farm of Alphas used for the Titanic movie), I'd be leery of using an entirely automated installation script. I'd rather have problems upgrading a few machines at a time than have every one of my computers in an undefined state and the phone ringing off the hook.

Red Hat 5.0 Tweaks and Tips

New Features

With “New VERSION 5.0” written in friendly large blue letters on the cover and a rewritten installation section, the manual has grown significantly since the 4.2 release. Release notes and upgrades are available on the Red Hat web site, as well as on the CD-ROM. Just as I wouldn't recommend seriously using NT or Windows 95 without the associated resource kit, I recommend a companion book like *Running Linux* or the *Linux Network Administrator's Guide* from O'Reilly and Associates.

Glibc and Linux 2.0.31—A Line Drawn in the Sand

The biggest change in 5.0 is the switch to a different C library, Glibc. Linux 2.1.x kernel development is hopefully nearing the end of a long and exceptionally chaotic development cycle—the new dcache, finer grained SMP and a raft of new features and performance enhancements are causing trouble. This is nothing new for Linux, but it's been a long time since the last “stable” release of 2.1. On top of all the ongoing kernel changes is a new standard GNU C library, Glibc. Glibc has a clean threads implementation, transparent support for IPv4 and IPv6, improved linking and other incremental improvements that ultimately turn it into a whole new animal. The combination of new library plus new kernel code makes it harder to isolate bugs.

By biting the bullet now, choosing to bundle the new C library and stick with a proven release of the kernel, Red Hat is doing the world's Linux developers and ultimately its users a service. Threading is a standard feature of competing operating systems, such as NT and Solaris, and very important to Linux's new bleeding edge GUIs: KDE and Enlightenment. In the short term there will be a flurry of Red Hat 5.0 incompatible software which should be cleaned up by the time you read this article.

The Not So Graphical Desktop

The graphical on-line help system is still hopelessly crude. The tools (notably **htdig**) exist to make an excellent searchable archive of all the included html/text documentation, and it's too bad that the default help system is so slow and clunky. The Red Hat 5.0 network configurator now has support for configuring a PAP (password authentication protocol) PPP session, which saves some troublesome scripting.

Security, Patches and Upgrades

Red Hat 5.0 includes many security enhancements. Because of the change to Glibc, all of the applications had to be recompiled and relinked. As a result, it is hard to tell which programs have been upgraded or changed.

Yet, there are (as of this writing) 54 additional patches in http://www.redhat.com/support/docs/rhl/rh50-errata-general.html and 8 in Intel, including a critical kernel upgrade that blocks the Teardrop attack. The Teardrop Denial of Service attack immediately made the kernel (2.0.30) shipped with this release obsolete. Someone decided to hit me with Teardrop, crashing my servers every night for two weeks. The newest kernel (2.0.32) successfully detects, logs and blocks this attack.

You should *immediately* install a new kernel and many of the RPMs if you intend to use your 5.0 system on the Internet. On installing a new release, the first commands I run are:

```
mkdir /usr/local/rpms
cd /usr/local/rpms
ncftp -R -d 5 \
        ftp.redhat.com:/pub/linux/redhat-5.0/
rpm -i -upgrade *
```

Substitute your favorite mirror for the FTP address. The **wget** utility (available from the GNU archives) is slightly better for mirroring sites in this fashion, as it supports both FTP and HTTP, partial file/directory transfers, time-stamped updates and automatically uses PASV mode (necessary for FTP to work through a firewall).

The fact that patches for these problems are so readily available is a tribute to the flexibility of RPM and the hard working bunch of Linux folk who are countering the persistent cracking community.

Maximum RPMs

RPM's FTP and FTP proxy support are now documented on the man page, but to use RPM effectively to distribute your own software requires the book, *Maximum RPMs*, available from Red Hat.

The UNIX world needs a software installer of the caliber of the Windows 95/NT Installshield. RPM is a good, even great start, but Debian Linux's installer beats the RPM/GLINT combination for both interactivity and ease of use. An RPM/GLINT release combining the best features of RPM and Debian would be a *good thing*.

Linux Versus Microsoft NT

Commercial Features

The RedBaron web browser has been dropped—Netscape 4.04 is freely available and is a vastly superior product. You will have to download it from the Net, however, as it's not bundled on the CD—yet.

Metro-X boasts of a nice configuration screen, added card support and improved color depth, resolution and performance over the XFree86 release of the X server. In my case, Metro-X is of marginal utility—the XFree and S.u.S.E. drivers for my ET6000 and Mystique cards are as fast or faster and more reliable. You have to configure XFree in order to even try to configure Metro Link in many cases.

Metro Link has some excellent add-on libraries, notably OpenGL, which is experiencing enormous interest in light of the PC gaming phenomenon. If you intend to use OpenGL, the bundled Metro-X server saves you money against the purchase of the extra OpenGL libraries. Your other choice is the freely available OpenGL clone, MESA, which converts OpenGL calls into X code—which is, in theory, much slower.

The new backup program with the unwieldy name of BRU 2000-PE is straightforward enough for a single-user environment. In my environment, where I have to back up multiple machines to a single tape drive, through a firewall, the combination of **cpio**/**dump** and **ssh** works exceptionally well.

The newly bundled RealAudio 5.0 server and client are very sexy products. I've now hooked up my studio to my main Linux box so that I can broadcast live sessions to the Internet. The server operated flawlessly. The live encoder worked great once I had a working sound card and worked out the correct encoding rate. The 95/NT versions of the software have nicer interfaces, but I've already had months and months of uptime on my primary RealAudio server.

Favorite Features

The sendmail featured in this release has improved anti-spam protection. No longer is it possible for a random spammer to use your host as an unwitting e-mail relay.

TheNextLevel is an excellent front end to X, offering much the same look and feel as Windows 95 and NT 4.0. The Red Hat 5.0 release of TheNextLevel is much closer to the look and feel of Windows 95; in particular, the “feature” of having to click to gain keyboard focus is enabled by default now, so first-time Linux users feel more at home.

Corporate Support

Red Hat has announced a set of support plans and partnerships that should make Linux more acceptable in the corporate world. Also, Linux's distributed support model has recently won awards from *Infoworld* and other magazines for “Best Technical Support”.

Conclusion

Over the course of these upgrades I did a number of things that are difficult, expensive or impossible to do with NT or Windows 95. Immediately after the base install was complete, I was able to exit the computer room. From the comfort of my own office, I completed the upgrades, added additional software, etc. From the comfort of my home, through *two* firewalls, I initiated a remote backup, did an A/B comparison of the new features of Red Hat via X in an ssh tunnel, wrote this review and installed, configured or updated even more of the Linux software. UNIX's enormous advantage in the remote administration area will probably continue into the age of high speed Internet access in the home.

Red Hat 5.0 is a solid core Linux distribution with a sharp eye out for the future. The new Glibc library implies a little risk and breaks some backward compatibility—expect to update this library a lot over the coming months as new bugs are found and fixed—however, the additional features are worth it. The future for both Red Hat and Linux looks very exciting—with the rapidly advancing KDE and GNOME/Enlightenment desktops and the release of Netscape source code.

But it's not all clover. During the installation, configuration and upgrade process there remain many problems which stopped two otherwise enthusiastic first time users and had me stumped more than once.

Due to the large number of patches and upgrades already required, I'm going to wait for 5.1 to put this release onto the rest of my production machines. I'm very concerned about potential security holes in Glibc. Red Hat 5.0 is an evolutionary release. Sometimes you have to take a step backward to take two steps forward. Red Hat 4.2 is a more mature, easier to use distribution than Red Hat 5.0. At this point I'd recommend 5.0 only to developers.

Red Hat is freely available from the Internet and in a commercial release that costs $49.95 US. It is also available bundled with books, manuals and/or applications.

Red Hat Alpha

Mike Taht is recovering from a two-year stint of running an ISP by writing, coding and surfing the Web from the comfort of his own ISDN line. He can be reached at mtaht@picketwyre.com.

Retro is into CGI/DBI programming, weird architectures and the EGCS project. He plans to disappear into the Santa Cruz Mountains with his laptop on March 31 and return with a working Netscape for Alpha Linux. He can be reached via e-mail at Retro@picketwyre.com.

Archive Index Issue Table of Contents

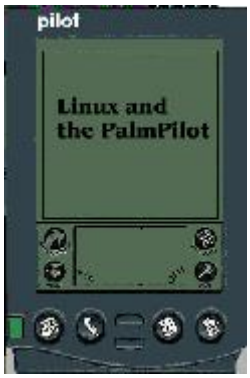
Advanced search

Linux and the PalmPilot

Michael J. Hammel

Issue #50, June 1998

This article contains all the information you need to run Linux on the PalmPilot personal digital assistant.



My brother and I share many things, not the least of which is an interest in electronic gadgets. I tend to go big—new computers, new monitors, new expenses. My brother thinks small—Personal Digital Assistants, also known as PDAs. He's collected a few different models over the past couple of years. Right now I think he's working on a PSION or possibly one of those Redmond atrocities known as Win/CE PDAs. My brother says of these that “wince” is about right.

I had mentioned sometime back that I wanted to get a PDA and thought a U.S. Robotics PalmPilot (now 3Com) would be the one to get. I had read somewhere that there was Linux software for use with the Pilot and that meant I could download and upload data directly to my Linux system. Fortunately, my little brother was listening that particular day. For Christmas this year he gave me his old PalmPilot. Okay, it's used—but it's still a cool gift.

The PalmPilot actually comes in four different configurations. The earlier versions are the Pilot 1000 (the version I now have) and the Pilot 5000. The newer versions are the Pilot Personal and the Pilot Professional. The older versions don't have a backlit display and come with significantly less memory.

They can, however, be upgraded. My brother had upgraded the 1000 to 1MB of memory prior to giving it to me. Besides more memory, the new models have version 2.0 of the operating system, and the Professional has a TCP/IP stack and e-mail tools.

Getting Started—The Linux Side Software

If you're going to use a PalmPilot with Linux, you need to start with its HOWTO document. The PalmPilot HOWTO is available from the Linux Documentation Project (LDP) as well as from its official site. The HOWTO, written by David H. Silber, is quite easy to follow and offers all the information you need to get started. I should note that I managed to get the software working using only the HOWTO—I didn't need to subscribe to any mailing lists or scan weeks of newsgroup archives. This is a credit to both the HOWTO author and the simplicity of the PalmPilot.

The HOWTO points the reader to the primary software archives from which you can download a variety of packages. The packages of interest to the new user are:

- **pilot-link**: The main package of interest, it includes the **pilot-xfer** program for transferring programs and data to and from the Pilot and Linux along with a host of other tools. The current revision of this software is 0.8.9. Note that the HOWTO links directly to the 0.8.2 version. The software archives list this package as pilot-link.x.y.z.tar.gz, where x.y.z is the version number.
- **PilotManager**: Graphical interface to the Pilot for UNIX/X systems. Handles most of the tasks that the pilot-xfer software does from the command line, but uses an X-based interface. This is a Perl/Tk application.
- **MakeDoc**: Turns text files into a document format better suited for the Pilot.
- **prc-tools**: The package you need in order to develop software for the Pilot.

The first package to retrieve is the pilot-link software, which contains a collection of about 30 programs. Most of them have very specific tasks such as uploading or retrieving “Memos” or “ToDo” lists. Of these programs, the most useful is pilot-xfer. This program is used to back up and restore complete programs and databases, even the entire Pilot system. It can even do incremental backups automatically. All of the programs in the pilot-link package are command-line programs.

PilotManager is a Perl/Tk graphical interface to the Pilot. It apparently works with CDE's Calendar applications, but since I don't run CDE I didn't try this feature out. The list of features states the PilotManager has four primary

features: uploading new applications and databases, synchronizing the Pilot with the CDE Calendar application, uploading text files as Pilot Memos, and backing up the Pilot to your UNIX system.

MakeDoc is a tool for converting ordinary text files into a compressed format. This format can be read by Rick Bram's PilotDOC reader, which is now a shareware product known as AportisDoc Trialware 2.0. Rick developed this reader because the Pilot's document handling was limited to relatively small documents.

Finally, prc-tools are a set of tools for building applications for the PalmPilot. This package is necessary only if you intend to do development work. If you go this route, you'll need to get the GNU **gcc**, **gdb** and **binutils** packages as well. All of this is explained in the HOWTO.

Building and Installing the Software

Building the pilot-link software is quite simple. After you've unpacked the package (**tar xvzf pilot-link.tar.gz**), decide where you want to install it. I installed it under /usr/local/pilot/. You must tell the configure script where the installation directory is; it assumes /usr/local. Should you also decide to install the man pages somewhere unusual, you should tell the configure script that as well.

To build the package, run the configure script, followed by **make** and **make install**. Here is the configure command I used:

```
configure --prefix=/usr/local/pilot\
    --mandir=/usr/local/man\
    --with-java=/usr/lib/java
```

The Java option was necessary to get a clean run in the configure step. It doesn't actually cause the Java extensions to be built—that has to be done manually, although these extensions are not required. You can use pilot-xfer and the other tools without the Java extensions.

After the make install was complete, I added /usr/local/pilot/bin to my path as:

```
PATH=$PATH:/usr/lib/pilot/bin
export PATH
```

At this point the pilot-link tools are ready for use with your Pilot. There are man pages for only a few of the programs in pilot-link, but most provide help if invoked without options or command-line arguments.

The Makedoc utility is a single C++ file. You can compile it using the following command line:

```
g++ -DUNIX makedoc7.cpp -o makedoc
```

Testing the Setup

The HOWTO doesn't spell this out, but it was rather easy to figure out after a short inspection of the code. Once you've built MakeDoc you should copy it into the same directory where you installed the pilot-link tools.

Once built, you need to test that the tools are working properly. The first thing to do is connect your PalmPilot HotSync cradle to your Linux box. To do this, connect the serial cable on the cradle to one of the serial ports on your computer. The cable on the cradle has a female connector, so you may need a gender changer to make the connection. I use an A/B switch box to connect both my external modem and Pilot to the same serial port (/dev/cua1) while using the other serial port for my mouse. This seems to work just fine, but be careful. If you switch between the two devices while one is running it can cause you to either have to manually reset the modem (turn it off and then on again) or reload your Pilot. So, be careful not to change the switch box setting while using either the modem or the Pilot.

Another thing you might want to do before trying the software is to make a symbolic link from the serial port to which the Pilot is connected to /dev/pilot. Most, if not all, of the tools in the pilot-link software use /dev/pilot as the default device, although they all take the port device as a command-line argument. Having the /dev/pilot symbolic link makes life a little easier.

You should also make sure that the serial device has the correct read/write permissions for whatever user ID you use while reading and writing to the Pilot. On my system /dev/cua1 has 666 permissions so that an ordinary user can access it. I don't know if this is a security violation or not, but it's my box—I'm the only user and I'm not too worried about security on it.

Now that the hardware is connected, you should test that the software can communicate with your Pilot. Place the Pilot in the cradle, making certain that it

is properly seated. (See the instructions that come with the Pilot.) The first test you should run is:

```
% pilot-xfer /dev/cua1 -l
```

where /dev/cua1 is the port the Pilot is connected to. If you aren't familiar with the serial port devices, /dev/cua0 is generally the first serial port and /dev/cua1 is the second. There may be others configured for your system. Once you've typed this command you will be prompted to press the HotSync button on the Pilot's cradle to begin the data transfer process.

The **-l** option for pilot-xfer will simply list the applications currently installed on the Pilot. If this works, you're in business. If not, check that the cable connectors are properly fastened together. Don't forget to press the HotSync button on the cradle once you've started the pilot-xfer command.

If the communication between pilot-xfer and the Pilot is working, you will get a list of the applications and databases installed on the Pilot. The results of this command vary depending on what else you have installed on the Pilot, but my stock configuration looks like this:

```
% pilot-xfer /dev/cua1 -l
 Waiting for connection on /dev/pilot (press
     the HotSync button now)...
 Connected
 Reading list of databases...
 'System Preferences'
 'Graffiti ShortCuts'
 'AddressDB'
 'MemoDB'
 'ToDoDB'
 'DatebookDB'
 List done.
```

Once you get this command working, do a backup of your Pilot data by typing:

```
pilot-xfer /dev/pilot -b `date +%Y%m%d`
```

I do this in my ~/lib/pilot directory. The “date” is actually the name of the directory where I want my backups placed. I use the format YYYYMMDD so that the directories sort correctly when listed using the **ls** command.

Now verify the programs and databases you just backed up:

```
pilot-xfer /dev/pilot -u `date +%Y%m%d`
```

The pilot-xfer program should recognize that the data has not changed and indicate that no changes are being made. That's what you want.

Pilot Programs and Databases

Pilot programs have a .prc extension. Pilot databases have a .pdb extension. Much of the HOWTO documentation refers to all of the files (programs and data) as “databases”. Unless you're going to do development work the difference isn't important. When you upload a .prc file to the Pilot, you've loaded a program that it will run. The .pdb file is the data the programs will use.

There are no user programs or databases included in the pilot-link package that you can upload to your Pilot. You can find these at one of the many shareware sites on the Internet. The place to start your search for Pilot applications is 3Com's official PalmPilot site: http://palmpilot.3com.com/, which includes links to a number of shareware and freeware sites. c|net also carries a collection of Pilot applications you can retrieve from their DOWNLOAD.COM site. This site also provides descriptions of the packages, which I find very helpful.

At the 3Com site, you will also find a collection of four games: Hardball, Minehunt, Subhunt and Puzzle. These are archived into a zip file. 3Com refers to this as a Windows file, but it can be used on Linux. Download it, taking care to specify a binary file transfer if you're using FTP. (Netscape and the other browsers should download it using a binary transfer automatically.) Once retrieved, use **unzip** to unpackage it. You should have four .prc files (file names are in uppercase) and a README.TXT. Be sure to read the text file. Now, let's take a look at how to upload these games to your Pilot using the pilot-xfer program.

Loading New Software and Databases

You've already made a backup, so you are now ready to try making changes to your Pilot. Change directories to wherever you unpacked the games.zip file and run:

```
pilot-xfer /dev/pilot -i *.PRC
```

The **-i** option is used to install new files into the Pilot. This command initiates the transfer of all four of the games in the package to the Pilot. The pilot-xfer program prompts you to press the HotSync button on the Pilot's cradle. Once you've pressed this button, the uploads will begin. Each of the applications uploaded will be listed by pilot-xfer. If you watch the Pilot's screen, you'll see the real name of the package as it is uploaded. This is the name the Pilot uses to reference the uploaded program.

When the upload is completed you should hear an audible signal, after which pilot-xfer exits. The programs are now ready to use. You can pull the Pilot out

of its cradle to check that the new games are there. Try them to make sure they are working properly.

That's all there is to it! See how easy it is to use the pilot-link software? As well written and easy to use as so much of the software for Linux is, I am still amazed and impressed at the true simplicity of the Pilot software.

Applications

Now it's time to look around for some decent applications. Below is a list of some programs you may wish to look into:

- HackMaster (shareware)
- AppHack (shareware)
- DinkyPad (shareware)
- ClockHack (freeware)
- LaunchPad (shareware)
- SelectHack (shareware)
- utilBas (freeware)
- Pilot Stopwatch (shareware)
- SketchPad (freeware)

All of these programs are available via links from the DOWNLOAD.COM site, as well as from a number of other Pilot shareware sites.

I highly recommend the HackMaster tool. It is needed in order to use the AppHack, MenuHack and a few others. The AppHack allows you to use the four real buttons (as opposed to the “silk screen” buttons) on the Pilot to launch applications. The MenuHack allows you to bring up the menu bars for applications by clicking on the top line of the window. Normally you need to click on the Menu silk screen button. LaunchPad is also a neat application. It provides a set of user-definable notebook tabs in which you can place the application icons. Consider it a file manager—not bad for a tiny device like the Pilot.

MakeDoc

MakeDoc is a Linux-side tool for converting text files into a format that can be used by the AportisDoc reader. To use MakeDoc, run it with a command that looks similar to the following:

```
makedoc data.txt data.prc "Data to display"
```

where data.txt is the name of the text file you want to convert and data.prc is the name of the file you will create. The text enclosed in double quotes will be used as the document title in AportisDoc's list of documents. Again, once you've converted the text file to the proper Doc format, you can upload it with pilot-xfer as

```
pilot-xfer /dev/pilot -i data.txt
```

Managing your Data

Once you've begun using your Pilot on a regular basis, you will want to do backups to your Linux box. Referring to the backup command we ran earlier,

```
pilot-xfer /dev/pilot -b directory-name
```

will do a complete backup of your Pilot. To do an incremental backup, give the command:

```
pilot-xfer /dev/pilot -u directory-name
```

The incremental backup should be done on an existing directory, but the complete backup should be done into a new directory. Like managing your Linux box, a good regimen of complete and incremental backups is recommended. Currently I do complete backups weekly and incremental backups whenever I add important data I don't want to lose.

Restoring lost data is fairly straightforward as well. The command

```
pilot-xfer /dev/pilot -r directory-name
```

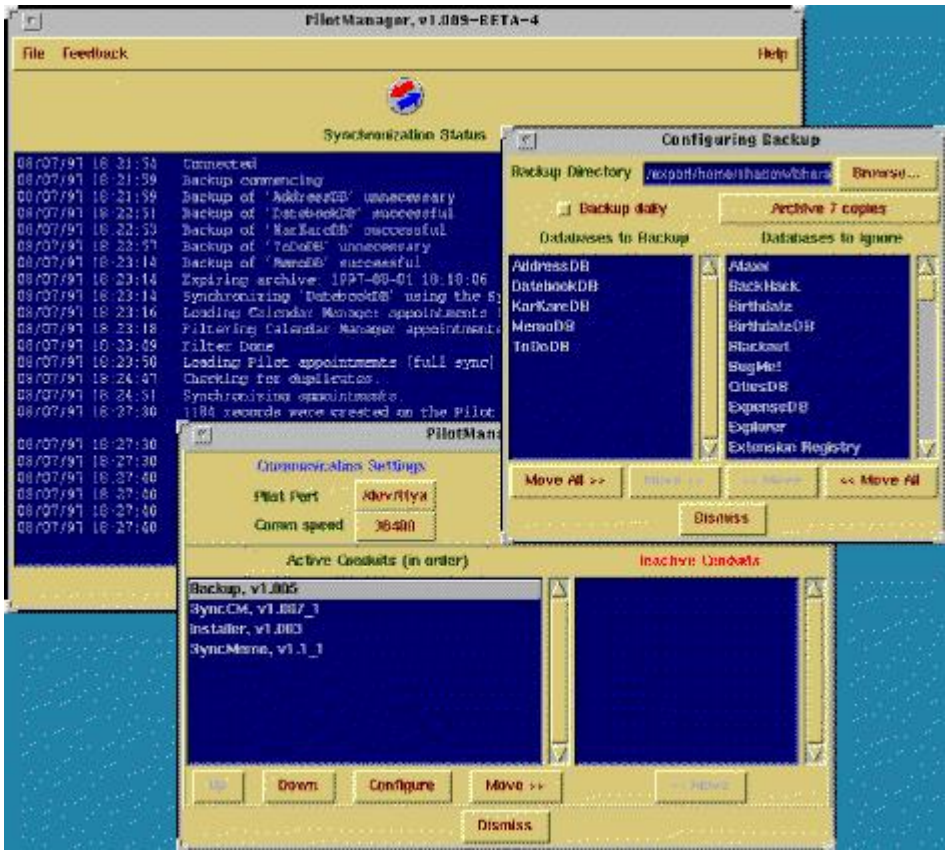
will upload the backed up data from directory-name to your Pilot. Pretty simple, isn't it? The pilot-xfer program can also delete and merge databases and purge unwanted data and programs from the Pilot.

PalmPilot Software and Linux

Resources

Summary

U.S. Robotics' PalmPilot has an advantage over other PDAs because of its open architecture, which means it's easier for the general public to create applications for this PDA than for some of the other popular PDAs. Because of this openness, there are quite a few freeware, shareware and commercial applications available for the Pilot.



Pilot Manager Screen Shot

The PalmPilot software available for Linux may have the best configuration-to-usefulness ratio of any Linux software. It's a breeze to get running and makes management of the Pilot from your Linux box, or other UNIX systems, very simple. Since the tools are simple in nature they can easily be incorporated into other tools. This makes the PalmPilot a terrific addition to the ever-increasing range of tools available for your personal and business use of Linux.

Michael J. Hammel is a software engineer currently working on network management tools for Samsung Telecommunications America in Richardson, Texas. Mike's interests run the gamut of technical to athletic, with a beer or two thrown in for social measure. Visit his home page at http://www.graphics-muse.org/ and e-mail him at mjhammel@fastlane.net.

Archive Index Issue Table of Contents

Advanced search

Administering Usenet News Servers

Liam Greenwood

Issue #50, June 1998

Administering Usenet News Servers claims to cover INN, DNEWS and PGP software and the planning, building and managing of Internet and Intranet news services, and it does.

- Authors: James E. McDermott and John E. Phillips
- Publisher: Addison-Wesley Developers Press
- E-mail: info@awl.com
- URL: http://www.awl.com/
- Price: $39.95 US (includes CD-ROM)
- ISBN: 0-201-41967-X
- Reviewer: Liam Greenwood

The world is sorely in need of a good book on Usenet news. Unfortunately, this isn't it. *Administering Usenet News Servers* has a definite lack of a clear focus. If the authors had a more tightly targeted audience, I believe this could have been an excellent book. Instead, it is self-defeating by trying to be all things to all news administrators.

What Does It Include?

Let's have a look at what it does cover. *Administering Usenet News Servers* claims to cover INN, DNEWS and PGP software and the planning, building and managing of Internet and Intranet news services, and it does.

The book discusses INN and DNEWS and covers installing and configuring both these NNTP news servers. The differences between the two are discussed, including why you might choose one over the other. Included are good appendices with DNEWS and INN configuration details. All you need to get an INN news server up and running under Solaris is included on the CD, as well as tools such as gcc. There is a clear walk-through of an INN install, with configuration variables that may need to be changed clearly marked. The installation information includes the PGP/INN configuration needed for managing newsgroup control messages.

Administering Usenet News Servers also has a reasonable discussion on the need for an “Acceptable Use Policy” (AUP) and what such a policy should address. It also makes some good points on the differences between an Intranet AUP and an ISP (Internet Service Provider) focused AUP.

There is a section on newsreaders, which mentions the good net-keeping seal of approval, but unfortunately only summarises the main points without including a pointer to the full text on the web.

There is a section on project planning, but it keeps a reasonable balance, showing that there should be a plan and mentions the sorts of things you need to consider, but it stops short of trying to teach you to be a project manager. The book also includes good coverage of some of the business issues involved with rolling out a news service on an Intranet.

The focus of the book is its weakness. It is very fond of Solaris as the example operating system to the extent that a chapter on installing Solaris is included. The binaries on the CD-ROM are for Solaris, and the book's examples assume a Solaris file system layout. Not providing Linux and *xxx*BSD binaries seems an oversight, but even if the authors wished to stay in the commercial arena, they should have considered covering more than just Solaris. I am left with the feeling that the title should have included a “For Solaris” tag.

The book seems to be mainly aimed at the news service being rolled out in an Intranet situation, but then leaps off into ISP land. However, it doesn't cover news to the depth needed by an ISP. This is demonstrated by the total dismissal without any discussion of both Cnews and UUCP. Even if the authors wished to recommend NNTP as the preferred news transport, both Cnews and

UUCP are still widely used and any self-respecting news administrator needs to know at least how to interact with Cnews systems and what effect UUCP has on things such as news propagation.

What about the quality? Did I find any major flaws? Well, PGP is not on the CD-ROM. The book says it is on page 121, while on page 221 it indicates you have to use FTP to get it. I also couldn't find DNEWS on the CD-ROM. The cover says that all you need to set up an INN or DNEWS service is on the CD-ROM, while the text of the book indicates you have to download a trial version of DNEWS. Furthermore, it misses some “gotchas” such as using DNEWS behind a firewall, where the DNEWS host that collects the news isn't always the one to which your clients connect. This can cause problems with the news-on-demand style of fetching news by DNEWS. (News-on-demand is the news delivery mode that DNEWS uses where no articles in a newsgroup are fetched from the upstream news providers until one of the local news clients attempts to read that group.) The CD-ROM also has some incorrect references, e.g., the book refers to the gcc-2.7.2 directory in its instructions—the actual directory provided is gcc-2.7.

Overall the book is nicely laid out and straightforward to read. My conclusion is that while it's not as comprehensive as it claims to be, it is a good overall view of what is required to create a functional news service. It will enable a UNIX-aware administrator to set up a vanilla NNTP service fairly painlessly. I think it would have worked better aimed squarely at the Intranet market, including binaries for AIX, Solaris and SunOS.

Liam Greenwood is an ex-librarian, ex-COBOL programmer, ex-Sperry system programmer and ex-UNIX Systems Engineer—now a Solutions Architect. In real life he's been running Linux since 0.99 when a friend found him the bits he needed to finish building a PC, so he could help support his Linux system. He can be reached via e-mail at liam@sasquach.gen.nz.

Archive Index Issue Table of Contents

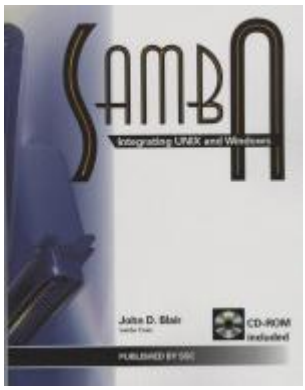
Advanced search

SAMBA: Integrating UNIX and Windows

Dan Wilder

Issue #50, June 1998

I can say with great enthusiasm that Samba is a fine package—proof that free software can achieve commercial-or-better quality.



- Author: John D. Blair
- Publisher: Specialized Systems Consultants, Inc.
- E-mail: info@linuxjournal.com
- URL: http://www.ssc.com/ssc/samba/
- Price: $29.95 US
- ISBN: 1-57831-006-7
- Reviewer: Dan Wilder

Samba, a subject of previous articles in *LJ*, is a software package that lets a UNIX or Linux system provide file and print services for Windows clients, including Windows for WorkGroups, Windows 95, Windows NT, and OS-2 and, for that matter, UNIX using **smbclient** or Linux using that or the Linux SMB file system. It earns lots of interoperability points for Linux, allowing a Linux system to earn its keep as a file server in a Windows shop, instead of the more costly, and perhaps less robust, Windows NT server.

For some years I have administered such a Samba installation. From that experience, I can say with great enthusiasm that Samba is a fine package—proof that free software can achieve commercial-or-better quality. Once installed, it is simple and easy to use; so much so, I've even seen Samba used to share user home directories between Linux hosts. Unfortunately, the ones who witness my enthusiasm for this software too often say, “Yes, it sounds great, if only I could manage to configure it.”

Samba configuration, especially the initial configuration, is not always a breeze. While sample configuration files, offering good starting points, are provided in the source tree, there remain many parameters to set or tune. These parameters reflect in part the significant complexity underlying Microsoft networking, and in part the difficulties of mapping the Microsoft file system and security model to the UNIX models, with their corresponding wealth of alternate solutions that offer various tradeoffs.

There is a lot of information in Samba's man pages and source tree /docs directory. However, this information isn't highly integrated, notwithstanding some nice HTML pages, and it is by no means complete.

When I first installed Samba, there was less information than now but it was still substantial. It took me a week of browsing through it, off and on, before I was ready to attempt my first Samba configuration. I remember thinking several times as I struggled to master this information and the gaps in it, “I wish there was a good book on Samba.” Now there is.

Written by John D. Blair, who has been made a member of the Samba team, all the snippets of information in the Samba source tree are combined in a single volume. (The Samba Team helped out by providing last minute information and editing.) More, lots of information that never found its way to the source tree is collected here. For example, Chapter 2 gives an introduction to SMB (service message block networking protocol), including some discussion of protocol details. This chapter contributes much toward understanding the context in which Samba operates.

Well-organized and well-indexed, this book guides the reader through a UNIX-eye tour of Windows networking concepts, terms and history, past the unpacking and building of the package, and into the complexities of configuring, testing and troubleshooting an installation. The Table of Contents, nine pages long, contains enough detail to avoid many trips to the extensive index.

You will find discussions of network service browsing, name resolution, performance, authentication and access control, name mangling, multiple

subnets, a very thorough chapter on problem diagnosis, and a whole chapter at the end devoted to the SMB file system.

Chapters 5 and 6 talk about the configuration parameters found in the main Samba configuration file, /etc/smb.conf. Perhaps the most important chapters for the working system administrator, each begins with an outline and its own index. This is a nice touch and potentially saves a lot of flipping back and forth to the main index or the Table of Contents. Options are grouped in these chapters by goal, rather than alphabetically as in the man pages, helping the reader draw connections between related options. Much of my week of browsing, when I first installed this software, was devoted to drawing connections between options.

The book comes complete with a CD-ROM containing Samba-1.9.18. This may be helpful for evaluating Samba; however, if you find this software useful, you should check the web site. At the time of writing this review (March), Samba-1.9.18p3 was current. No doubt a few more patches will be released between now and the time you read it.

Mr. Blair apparently felt obligated to include information on using both 1.9.17, which required the U.S. export-controlled DES library, **libdes**, and 1.9.18, which includes exportable limited-DES hashing code. You may already be familiar with the peculiar U.S. laws regarding export of encryption software. Even encryption software featuring the venerable and well-known DES algorithms, software written outside the US and freely available around the globe, is affected by these laws.

The Samba team has found a solution that may avoid the issue. Samba now includes the DES algorithm, in the form of a dedicated implementation used only to calculate a hash value as its output—this value cannot easily be deciphered to provide the original input. This method is apparently exempt from classification as encryption software. In any case, there are no obvious export restrictions on the present version, and the necessity of obtaining a DES library to link Samba against is avoided.

Having encryption support is nice, as it avoids multiple re-entry of passwords when signing onto NT. Without it, Samba shares that are marked for automatic network mount on NT login will result in your being asked to enter your password repeatedly, a situation that becomes quite frustrating. With encrypted password support, if you use the same Samba password on the server and locally on your NT system, you'll be prompted for your password only once when you sign on to NT.

The book covers the use of libdes with Samba-1.9.17, at some length, in several places. In each, there are words which sound like afterthoughts, to the effect that none of this is necessary with 1.9.18. I'd rather have seen this 1.9.17 information moved out of the way to an Appendix.

This is a minor blemish, however. I have enjoyed reading this book in its galley proofs, and it has helped me in a recent upgrade of several Linux systems from Samba-1.9.16 to the current version. I look forward to obtaining this book in its final printed form.

I believe *SAMBA: Integrating UNIX and Windows* will be of great assistance to those who like to get started with a good book. Use of the Samba software, combined with the information contained in this book, will enable Linux to continue to establish a foothold in mostly-Windows shops for use as a file server.

Dan Wilder writes software and book reviews in Seattle, Washington, where he also tends a garden and raises two wonderful children with his wife, Jacque. Dan can be reached by e-mail at dan@gasboy.com.

Archive Index Issue Table of Contents

Advanced search

Server-Side Includes

Reuven M. Lerner

Issue #50, June 1998

Don't want to learn CGI but still want dynamic web pages? Mr. Lerner introduces us to server-side includes.

Most web sites contain largely static HTML files jazzed up with some images. More advanced sites use one or more CGI programs, HTTP cookies, database backends and other topics we have discussed in previous months.

All of these techniques involve creating at least one new process on the web server. If we are interested in keeping our server's load as low as possible, we should avoid creating unnecessary processes. I am not suggesting we remove CGI programs—but I am saying there are times when CGI might be overkill.

Sometimes, for instance, you can do everything you need with server-side includes, also known as “SSIs”. SSIs offer a good balance between efficiency and complexity. Even if you have never seen them, server-side includes are pretty easy to understand. Indeed, they are often ideal for giving non-programmers a taste of dynamic page output without confusing them with the problems associated with actual programming. Here is one example of what they look like:

```
<!--#printenv -->
```

Yes, this looks like an HTML comment. But unlike an HTML comment, which is passed along to the user's browser unmodified, SSIs are parsed by the server before the file is sent off. It might be easier to think of SSIs as macros that are expanded by the web server.

Each server-side include begins with the “open comment” characters (**"<!--"**), a hash mark (**"#"**) immediately following the two dashes, the name of the command you wish to evaluate, whitespace, zero or more attribute value pairs followed by whitespace and finally the “close comment” characters (**"-->"**). So

the SSI command **#printenv**, which takes no arguments and returns the list of environment variables, could be contained in the following file:

```
<HTML>
<Head><Title>Testing</Title></Head>
<Body>
<H1>Testing</H1>
<!--#printenv -->
</Body>
</HTML>
```

Before sending the above document to a user, it would be expanded into something resembling Listing 1. Notice how our simple SSI was replaced by a list of environment variables and their values. While there is no standard for server-side includes, servers share a common list of SSIs with each server defining new ones.

This month, we will look into server-side includes—from configuring your server to allow for them, to some different SSI commands you might wish to use on your web site, to a number of ways in which to use SSIs on your site.

Configuring Apache for SSIs

Before you can actually create pages containing and using server-side includes, you must configure your web server to allow for them. If you are compiling Apache from scratch, make sure that **mod_include**, the module that takes care of SSIs, is compiled into the server. (By default, it should be.)

Even if mod_include exists, you must configure several additional items. First, you must tell Apache you wish to allow SSIs by using the **Options** directive in the server configuration file.

On my Red Hat 4.2 system, the file containing this information is called /etc/httpd/conf/access.conf, and the line looks like:

```
Options Indexes FollowSymLinks Includes
```

This indicates that I have decided to activate three of Apache's options—**Indexes** (producing a directory listing if a user asks for a directory rather than a file), **FollowSymLinks** (telling Apache to follow symbolic links, rather than ignoring them), and **Includes** (meaning server-side includes should be active).

If you would prefer to stop users from using **#exec**, which allows them to run arbitrary external programs, replace **Includes** with **IncludesNOEXEC**, as follows:

```
Options Indexes FollowSymLinks IncludesNOEXEC
```

If you wish to allow SSIs in only one directory, modify the configuration file so that the **Options** line appears between **<Directory>** and **</Directory>** lines. For

example, if we only want files in the /ssi directory to allow for server-side includes, we could give this:

```
<Directory /home/httpd/html/ssi/>
Options Indexes FollowSymLinks Includes
</Directory>
```

The Apache documentation also describes how you can have several **<Directory>** blocks. If your server hosts several sub-sites, you can use them to define different services for different sub-sites, depending on who is running them, how much they have paid or what policies you wish to have in place.

We indicate which files might contain SSIs by using two additional directives in the srm.conf configuration file. The first, **AddType**, indicates what sort of content-type header should be sent when the server returns a document with an .shtml suffix. Browsers need to know how to interpret the data being sent to them—it could, after all, be an image in JPEG format, text in HTML format or completely unformatted data. We thus add the following line to our configuration file:

```
AddType text/html .shtml
```

However, that's not quite enough; we also want files to be parsed by the server on their way out the door. This is done by instructing Apache to use the “server-parsed” handler on all files with “.shtml” as the suffix. We can do this by adding the following line to the srm.conf file:

```
AddHandler server-parsed .shtml
```

and then restarting the server.

You might be wondering why we must use .shtml, rather than .html. Why not add a server-parsed handler for .html, and dispense with the separate extension?

The answer has to do with server efficiency. SSIs have less computational overhead than CGI programs, but *less* is not *none*. If we were to tell Apache that all HTML files might include SSIs, Apache would have to inspect every .html file, which might slow things down significantly. Thus, it is customary on many sites to divide files into two categories—those containing only HTML (with the .html suffix), and those containing HTML plus server-side includes (with the .shtml suffix). The only difference to users is the file extension they eventually see, since SSIs are replaced by their results before they are sent to the user's browser. Both are sent with a content-type of “text/html”, since our **AddHandler** directive instructs Apache to do so.

Using Server-Side Includes

Now that we have told Apache how to handle SSIs, we can start to use them in our files. Once again, only those files with .shtml suffixes will be parsed by Apache's server-side include mechanism, so make sure to save your files with an .shtml suffix, rather than with an .html suffix.

As we saw above, server-side includes look like HTML comments. This means any server-side includes not parsed by Apache will be invisible to the end user. Even if the SSI is passed unmodified to the user's browser (because of an error or a misconfigured server), there won't be any problems or oddities in what the user sees.

One of the most common uses of SSIs is to indicate when a document was modified. This is useful when a page is updated regularly; a typical example might be a news service or events calendar.

Here is a file indicating its latest modification date. We print the date with the SSI **#echo** command, which prints the value of an SSI variable. SSI variables include environment variables, plus several others defined by Apache. In this example, we look at **LAST_MODIFIED**, which contains the date and time of when the file was changed:

```
<HTML>
<Head><Title>I was modified</Title></Head>
<Body>
<H1>I was modified</H1>
<P>I was last modified on
<!--#echo var="LAST_MODIFIED" -->
</P>
</Body>
</HTML>
```

If you have followed the instructions so far, retrieving this page should indicate when it was last modified. Remember to save the file with an extension of .shtml—while writing this column, I spent some time trying to figure out why one particular SSI wasn't working. As it turns out, the problem was with the file extension, not my server.

The date printed by **#echo** might look nice to programmers, but it is a little daunting for most people. Non-programmers would prefer a slightly more familiar date and time format.

Fortunately, Apache allows us to modify the way in which dates are displayed. C programmers are probably familiar with the **strftime** function, which allows for the creation of many different time and date strings by using characters preceded by percent signs (%). For example, “%A” gives us the name of a day, “%B” returns the name of a month, “%d” gives the day of the month and “%Y”

returns the four-digit year. Thus by specifying “%A, %d %B %Y”, we can get a string that looks like “Wednesday, February 22 1998”.

Here is an example of setting the date to American format, first using the “config” SSI, and then using the “echo” SSI to display the results in our new format.

```
<HTML>
<Head><Title>Testing</Title></Head>
<Body>
<H1>Testing</H1>
<!--#config timefmt="%m/%d/%y" -->
<P>In America, I was changed on
<!--#echo var="LAST_MODIFIED"></P>
</Body>
</HTML>
```

Already, we can see a pattern in how server-side includes are defined and used. They consist of a keyword and then one or more attribute,value pairs, just as in this example:

```
<HTML>
<Head><Title>Testing</Title></Head>
<Body>
<H1>Testing</H1>
<!--#config timefmt="%m/%d/%y" -->
<P>In America, I was changed on
<!--#echo var="LAST_MODIFIED"
--></P>
<!--#config timefmt="%d/%m/%y" -->
<P>In Europe, I was changed on
<!--#echo var="LAST_MODIFIED"
--></P>
</Body>
</HTML>
```

Importing Other Files

Printing a file's modification date is fine if you are running a continuously updated news service, but it does not have many other applications. By contrast, I find the file-inclusion SSI functions are extremely useful when designing sites.

The syntax is quite simple, as you can see from this example:

```
<HTML>
<Head><Title>A basic template</Title></Head>
<!--#include virtual="/fragments/header.htmlf"
-->
<P>This is the text of my page, sandwiched between
two server-side includes.</P>
<!--#include virtual="/fragments/footer.htmlf"
-->
</Body>
</HTML>
```

Here, we use **#include**, with a single argument named “virtual”. Apache replaces the contents of this SSI with the contents of the named file. This might not seem all that useful, but consider how easy this makes it to create a site with a

uniform look. The header.htmlf fragment could contain the standard **<Body>** tag, defining text and background colors, as well as putting a menu bar across the top of the page. By the same token, the footer.htmlf fragment could contain a copyright notice, smaller menu bar or information about the server.

What is the advantage? When you decide to add a new button to the menu bar or when the site's sponsors move to a new address, you only need to modify a single file. The changes propagate automatically through the rest of the site. This is easier than changing each individual page, and more efficient than creating the page with a CGI program. Just as you can avoid programming errors by putting repeated instructions into a subroutine, so too you can avoid typos and other potential problems by putting repeated information into HTML fragments that are imported with the “include” SSI.

Tips and Tricks When Including Files

If your site uses CGI programs to create dynamic pages, you might be tempted to include your standard headers and footers in the programs' output using **#include**. Unfortunately, because CGI programs and SSIs use different handlers, there isn't any way for this to work. If you decide to use HTML fragments as headers and footers, you might want to define some short subroutines that can be included in your CGI programs.

Also, because different handlers are used for SSIs (“server-parsed”) and CGI programs (“cgi-script”), you cannot include server-side includes in the output from CGI programs and expect them to be interpreted. If you decide to create a uniform look and feel for your site using HTML fragments (described below), any CGI programs you write will be able to include those fragments. If you write CGI programs in Perl, such a subroutine could look like Listing 2. Your CGI programs would then look like Listing 3. Now when you change header.htmlf or footer.htmlf, all output on the server—from HTML files and CGI programs alike—will immediately reflect the changes.

In case you are wondering, fragments are imported verbatim, and any SSIs they might contain are passed along as HTML comments. Assume we defined header.htmlf to be the following two-line fragment:

```
<P>This is the header.</P>
<!--#printenv -->
```

If this fragment were retrieved directly through Apache, the **#printenv** SSI would print the current list of environment variables. But since header.htmlf is imported via a **#include** SSI, the **#printenv** function is sent to the user's browser uninterpreted. This might seem unnecessary, until you consider that allowing

SSIs inside of included files might lead to infinite loops or other unexpected results.

Variables and Conditionals

One of the more interesting recent additions to server-side includes is a limited programming language allowing for the setting and testing of variables.

Setting variables is fairly simple; you can do it with the following syntax:

```
<!--#set var="varname" value="value" -->
```

You can see the results with **#echo** (for a specific list of variables) or **#printenv** (for all defined variables), as in the following example:

```
<HTML>
<Head><Title>Setting variables</Title></Head>
<!--#set var="pi" value="3.14159" -->
<pre><!--#printenv --></pre>
<P>pi = <!--#echo var="pi" --></P>
<HR>
<!--#set var="e" value="2.71828"
-->
<pre><!--#printenv --></pre>
<P>e = <!--#echo var="e"
--></P>
</Body>
</HTML>
```

The above example also demonstrates how SSIs are interpreted in the same order as they appear in the file. The output from **#printenv** changes after each variable setting.

Setting variables is useful when used in conjunction with **if-then** statements. These statements can be used to create conditional text within HTML files without having to use CGI programs. The syntax is rather simple, for example:

```
<!--#if expr="$SERVER_PORT=80"
-->
<P>You are using server port 80</P>
<!--#else -->
<P>You are using a non-standard server port</P>
<!--#endif -->
```

Note that the variable name in an **#if** statement must be preceded by a dollar sign, much as with shell scripts. The **#else** statement is optional, but the **#endif** is mandatory, indicating the end of the conditional text.

You can even perform pattern-matching within variables, using regular expressions, as in the following:

```
<HTML>
<Head><Title>Browser check</Title></Head>
<!--#if expr="$HTTP_USER_AGENT = /^Mozilla/"
-->
```

```
<P>You are using Netscape</P>
<!--#else -->
<P>You are using another browser</P>
<!--#endif -->
</Body>
</HTML>
```

If the value of HTTP_USER_AGENT (normally set to a string identifying the user's browser) is set to

```
Mozilla/4.04 [en] (X11; I; Linux 2.0.30 i586; Nav)
```

as is the case on my system, the above will evaluate to “true”, and thus print the first string. Otherwise, it will print the second string. In this way, you can create menus customized for each browser. For instance, you could make life easier for users of Lynx (a text-only browser) by giving them a separate menu structure that does not rely on images.

Conclusion

Server-side includes do not solve all problems—but what software does? Rather, SSIs were created so that non-programmers could create dynamic output. Over time, they have expanded to the point where they can now include conditional statements, which are a first step toward actual programming. As we have seen, though, programmers can benefit from many of SSI's features, especially when it comes to including simple information inside of pages of HTML, such as standard headers or a file's last modification date.

There are a number of other commands available from within SSIs, including **#exec**, which allows you to run a program and incorporate its output into a page of HTML. (You can also use **#include** to bring in the output from a CGI program, even if you use **IncludesNOEXEC** rather than **Includes** in the Apache configuration.)

In some cases, though, such simple server-side includes might not be enough. Over the next few months, we will look at several software packages that take the idea of server-side includes one step further, making a complete programming language available inside of HTML files without the need for CGI programs.

Resources

Reuven M. Lerner is an Internet and Web Consultant living in Haifa, Israel, who has been using the Web since early 1993. In his spare time, he cooks, reads and volunteers with educational projects in his community. You can reach him at reuven@netvision.net.il.

Archive Index Issue Table of Contents

Advanced search

Letters to the Editor

Various

Issue #50, June 1998

Readers sound off.

Attaching Files to Forms

Thank you for the article by Reuven Lerner. I have been using file uploads for some months to allow students to upload assignments to my site for marking. Since they already have accounts, I use their UNIX password to check who they are.

File upload is part of HTML 3.2 and *should* be available from every browser. However, the standard says this is a slightly obscure part and may not always be supported. Internet Explorer 3 is one browser that does not support it, for example. Instead of a file selection box, the user sees a text entry area, and only the file name is uploaded, not the contents of the file. You need to guard against such browsers both at the HTML end ("You should see a file selection box here") and by checking the output at the server end.

—Jan Newmarchjan@ise.canberra.edu.au

Hmm, I have been using attached files for months with a variety of clients, and no one ever mentioned this. (I thought I had tried it with Internet Explorer. I know some of my clients use IE, but I guess they used Netscape or something when they uploaded files.) Thanks for teaching me something new.

—Reuven LernerReuven@netvision.net.il

S.u.S.E. Review

I regularly buy *LJ* here in Switzerland, and while in the USA recently I took the opportunity of buying Issue 46 a few days early as one of my particular interests is database design. While in general I found issue 46 was up to your

usual excellent standards, it was unfortunately spoiled for me by the following throw-away remark made by Stu Green in his review of S.u.S.E. V5.0:

> There are some minor errors in translation from the German, including the presence of some characters unique to that language being left as is, in particular in the names of individuals. These mistakes are easy to overlook.

Perhaps I have been living in Switzerland (a country with four national languages) too long, and I'm missing some ironic humour here. The alternative possibility, that Stu actually believes that people spelling their own names with characters from their mother tongue constitutes a *mistake*, is surely too insular even for Texas!

Given the international history of Linux (please note that most of the characters *unique* to German also exist in, for example, Finnish), it's a shame to see this Anglo-centric view of the world persist.

Perhaps the only crumb of comfort I can find in this situation is that software developers here in Europe and in Asia will continue to be able to market their products to several hundred million consumers with minimal competition from the English-speaking community.

—Paul Kennedy

Sybase Database for Linux?

In the February 1998 *LJ*'s "From the Editor" under Databases, you said, "Sybase sells an official Linux version but refuses to support it." I have spoken with several Sybase salespeople, and all of them told me that they do not sell any version of their SQL server for Linux.

I was wondering where you got this information, or better yet, who I might talk to in order to purchase a copy.

—James Pricejprice@dwwc.com

A system administrator, who was researching databases in order to buy one, made this statement to linux-list on-line. Since he was someone I know to be trustworthy, I believed it without checking. [Always a big mistake.] I have since learned that Sybase did have a client side freely available for a while, but work was stopped and the server side was not done. It is now being worked on once more. Our publisher Phil Hughes has been talking to the programmer doing the port. Sorry for the misinformation—Editor

Removing Files and Security

I'm writing in reference to the “Best of Tech Support” item entitled “How Do I Remove This File?” (March 1998).

The information given in response to the question is absolutely correct. However, beyond the basic information about how to delete such files, a warning should be added: discovery of such files is a bright red flag that your system may have been compromised by hackers.

Hackers will very often use file names and directories with such names as “. ”, “.. ” and “...”. These names are easy to ignore in a directory listing and are commonly overlooked by novice (and even experienced) users. Also, hackers will use directory names of legitimate applications, such as “.elm”, “.data” and “.tin”, because these directories don't show up in a normal **ls** listing and because they appear normal. Naturally, there are many variations on this basic theme, but if you spot such directories in unexpected places (or even in legitimate user directories), further investigation is definitely warranted.

Another warning flag is the presence of IRC files. The IRC is a seething hotbed of hacker activity these days, because it's so easy to become anonymous and because of the total lack of security controls inherent in the entire IRC system. So called “warez” channels provide an easy and totally anonymous way for hackers to exchange pirated software and hacking tools. If you start seeing “eggbot” files on your system, it's possible at least one of your user IDs is being misused. It's been my experience that many of these people want only to quietly misuse a stolen account for purposes of running their IRC bots, but some of them have attempted some really nasty attacks. In general, it's wise to cast a suspicious eye on any sort of unexpected IRC activity on your system.

Finally, another trick currently in use by hackers is to use **lynx** to download hacking tools. By storing their files on a web host and then using lynx to retrieve them, they can bypass the logging that often occurs with an FTP server and may be able to blend in more easily as a legitimate user.

—Dave Lutzdlutz@smith.edu

Thanks for GPIB Article

First I want to express my appreciation for the article “GPIB: Cool, It Works with Linux!” by Timotej Ecimovic, March 1998. I have worked with HPIB and GPIB, and I appreciated his treatment of that standard and Linux. I felt a special appreciation for what I perceived as his “spirit of Linux” or “spirit of GNU”. He appeared to me to express profound respect for the efforts of others. He also

impressed me with his technical honesty in that he did not attempt to portray Linux as the perfect solution for all situations.

Now a minor critical remarks—I believe that the “About the cover:” is in error in that the screen displayed on the cover is a snapshot of FVWM95, not FVWM (or FVWM2).

Linux Journal is one of the few magazines I not only *hold onto*, but actually *do* go back to past issues for reference. Keep up the excellent work—even your advertisements are of superior quality. *Linux Journal* is a “strange beast” for me as I almost never read advertising in a publication, yet I find myself reading just about all of the ads in *LJ*.

—Bill Leachbleach@BellSouth.net

KDE Comment

First of all, I would like to thank you guys for providing such an excellent publication. In regards to your March 1998 issue, I was surprised to see there was no article on KDE. I've used virtually all of the window managers that are available on the Internet and believe that KDE is the future of window managers. Why? KDE provides the ease of use of those other operating systems while at the same time utilizing the power of Linux/UNIX. Anyone who uses X should give KDE a try. Find it at http://www.kde.org/.

—Jeffrey Lojylo@ucdavis.edu

We did get a KDE article for March, but it was one of the last to arrive and the issue was already full. We will publish it in the near future. In the May issue of *LJ* there will be a short *Linux Gazette* column reviewing KDE and GNOME. Miguel de Icaza is writing an article about GNOME for us—Editor

Red Hat 5.0

I can't believe just how bad Red Hat 5.0 truly is. I've used Linux since about kernel 0.99 and have used Red Hat since version 4.0 and never have I come across such a terrible release. The bug list is incredible (check the errata page on their web site), and worse still, there aren't fixes for all of them yet. Rather than just post the patches, Red Hat seem intent on forcing RPMs down your throat by making users download multi-megabyte files rather than small patches that users could apply to the source code on the second CD and build themselves.

Well, Red Hat, this is a very Microsoft-like effort from you. Your own standards (RPM, AnotherLevel—heavily bugged) forced upon users, and an operating

system that will be a commercial success but a technical failure. On a final note, why won't you answer my e-mail for technical support when I'm a registered user who purchased the package?

—Simon Mauricemaurices@mpx.com.au

Setting Up E-mail

My hat is off to Jonathan Walther for an extremely useful article on Linux home e-mail. I had wondered for *years* how this was done. In the past I would start a PPP connection to my ISP, then use **telnet** to log in to the ISP in order to use their Pine to read the mail. Printing an e-mail involved exporting to a file on the ISP's server, dropping the TELNET connection, using **ftp** to transfer the file to my machine and, finally, printing the file.

But no more—thanks to Jon for turning on the lights. The whole process took less than an hour and went exactly as outlined in the article. Everything came up the first time I tried it; no problems with anything. And this e-mail is coming to SSC from *my* Pine on *my* machine in *my* house!

—Bill Cunninghambwc@coastalnet.com

Linux in Colleges

I just wanted to say how much I enjoyed March's article “Linux Means Business: Colleges Using Linux” by Don Kuenz. I'd like to see this view of colleges using Linux become a regular column—one for businesses and one for education and research. I know a lot of people would be interested in this type of article. For example, our school is very big on Linux. Our system is the COE Mosaic Linux Tile program which uses Linux machines connected to our existing AFS system. More information can be found at http://linux.uncc.edu/.

I consider this to be pretty impressive and something that other schools may like to see and try. At the same time I think we could learn a lot about other schools' implementation of Linux in the curriculum and for work purposes. We are very active with Linux at our school. Our newsgroup uncc.Linux is one of the most popular at the University, and we have a Linux users group that includes our trusty penguin mascot at each meeting. I enjoyed the article and hope to see more like it in the future.

—Brandon Perkinsbdperkin@uncc.edu

Issue #44, Worth Its Weight in Gold

For nearly a month I agonized and procrastinated over my need to develop a web site consisting of hundreds of pages that were similar and related. Creating a few pages is easy and fun, but anything more becomes tedious and boring. When I turned to the article “Industrializing Web Page Construction” in the December 1997 issue (#44), I learned that tools have already been developed to solve such problems. I installed the software and created 60+ pages in hours. Thank you *LJ* and Pieter Hintjens.

—Richard Parryrparry@qualcomm.com

Archive Index Issue Table of Contents

Advanced search

Connectivity

Marjorie Richardson

Issue #50, June 1998

To help you get connected, we have an article about a GUI that makes setting up PPP easy, as well as articles explaining how to set up PLIP, NFS, NIS and qmail.

We have a great issue for you this month. To help you get connected, we have an article about a GUI that makes setting up PPP easy, as well as articles explaining how to set up PLIP, NFS, NIS and qmail. I think you will find something of interest in each of them.

We are also fortunate to have an article from Peter Braam about the CODA file system with excellent illustrations by Gaich Muramatsu. Also, John Blair interviewed the Samba team for us to find out what's new in that arena.

The Silicon Valley Linux User's Group scored a coup by having Linus Torvalds as their guest speaker in March. Chris DiBona not only wrote it up for us, but sent along pictures too.

Next month also promises to be a great issue. Our focus will be Science and Engineering, and I think you may be surprised at some of the places Linux is being used to do research around the world. For example, Linux is being used in high-energy nuclear studies being done in Geneva by CERN and in ocean surface studies by the British Antarctic Survey.

Upcoming Events

A couple of events not to miss are coming up. Linux Expo looks to be bigger and better than ever. The list of speakers found at their web site, http://www.linuxexpo.org/, is quite impressive. Linux Expo will be held in Chapel Hill, North Carolina, May 28-30.

Then June 15-19 there will be the 23rd Annual USENIX Technical Conference held in New Orleans, Louisiana. Now there's a fun place to hold a technical conference. For details, see their web page at http://www.usenix.org/.

Lottery Problem

In our April issue, James Shapiro posed a problem concerning the best way to accept lottery winnings in his article “Financial Calculation Programs for Linux”. We feel we've kept you in suspense long enough, so the answer to that problem is included here (see sidebar).

Lottery Answer

Archive Index Issue Table of Contents

Advanced search

Open Source Summit

Eric Raymond

Issue #50, June 1998

The summit was hosted by O'Reilly & Associates, a company that has been symbiotic with the Open Source movement for many years.

On April 7th, 1998, a select group of the most influential people in the Open Source community gathered in Palo Alto to meet each other, consider the implications of Netscape's browser source release, and discuss where the Open Source movement is headed (and, especially, how it can work with the market rather than against it, for the benefit of both).

The summit was hosted by O'Reilly & Associates, a company that has been symbiotic with the Open Source movement for many years. Linux's own Linus Torvalds attended. The inventors of all three major scripting languages were present: Larry Wall (Perl), John Ousterhout (Tcl) and Guido Van Rossum (Python). Eric Allman (Sendmail) and Paul Vixie (BIND/DNS) were present, representing their own projects and the BSD community. Phil Zimmerman, the author of PGP, was there too, as was John Gilmore, a co-founder of Cygnus and the Electronic Frontier Foundation. Brian Behlendorf spoke for the maintainers of Apache. Jamie Zawinski and Tom Paquin represented Netscape and mozilla.org. For my semi-accidental role in motivating the Netscape source release with “The Cathedral and the Bazaar”, I also had the honor to be among those invited.

We met from 8:30AM to 5PM, following up with a well-attended press briefing. It was invigorating just to be around the amount of intelligence and accomplishment there, and a bit sobering to realize how absolutely critical their work has become—not just to the hacker culture but to the world expecting the Internet to become the vital communications medium of the next century.

One of the most important purposes of the meeting was simply to permit everyone to meet face to face, shake hands, look in each others' eyes and hear each others' voices. Many of us had never actually met each other before,

despite having been in e-mail conversations for many years. Tim O'Reilly felt (correctly, I think) that Net contact has not been quite enough as a community builder; that the opportunities and challenges we face now require an attempt to build more personal trust among the chieftains of the major Open Source tribes.

In that, I think, the meeting was very successful. But it also certainly dealt with substance as well. We discussed different perspectives on the Open source/ free software phenomenon and different definitions of it. One of the meeting's important results was a general agreement that, in all the variant definitions, *public access to source* was the most important and only absolutely critical common element.

We discussed the vexing issue of labels, considering the implications of “freeware”, “sourceware”, “open source”, and “freed software”. After a vote, we agreed to use “Open Source” as our label. The implication of this label is that we intend to convince the corporate world to adopt our way for economic, self-interested, non-ideological reasons. (This is the line of attack I've been pursuing though www.opensource.org and many recent interviews with the national press.)

We talked about business models. Several people in the room are facing questions about how to ride the interface between the market and the hacker culture. Netscape is approaching this from one side; Scriptics (John Ousterhout's Tcl company) and Eric Allman's commercial Sendmail launch are approaching it from the other. No one is certain yet what will work, but we were able to identify common problems and some possible strategies for attacking them.

We talked about development models—the various ways in which projects are organized, the strengths and weaknesses of each model, and what our individual experiences have been. There were no magic insights, but again it seemed helpful to recognize common problems.

We all understood this meeting could be only a beginning. Late in the day we developed a tentative agenda for a larger follow-up conference which O'Reilly may host later in the year. We hope to bring other key people from the Open Source community in on that follow-up—one of the last things Tim asked us to think about was who should have been with us, but was not.

The day ended with a well-attended press briefing at which all of us answered questions from Bay Area and national reporters—some got the message, some didn't. For every one who genuinely wanted to understand the logic of the Open Source approach, there was another who repeated “let's-you-and-him-

fight” questions about Microsoft. Still, the first burst of publicity about our gathering (it is two days later as I write) has been very positive.

We are entering a very exciting time. In the wake of the Netscape release, the Open Source community has achieved a visibility it never had before. We're making friends in new places and meeting new challenges. The larger world we're now trying to persuade to adopt our way doesn't care about our factional differences; it wants to know what we can do for it that is valuable enough to motivate a major change in the ground rules of the software industry.

To do that persuading, we'll need to pull together as one community more than we have in the past. We—not just the Linux community but the BSD people, the Perl, Python and Tcl hackers, the Internet infrastructure people and the Free Software Foundation—will need to present one face and speak one language and tell one story to that larger world.

That is, ultimately, why this meeting was so important. All of us came away with a better sense of what that story is and how each of the major tribes fits into it. Just the fact that we faced the reporters (and, by extension, the rest of the world) together was a very powerful statement. The summit was a good beginning—one to build on in the coming months.

Eric S. Raymond is a semi-regular *LJ* contributor. You can find more of his writings, including his paper “The Cathedral and the Bazaar”, at http://www.ccil.org/~esr/. Eric can be reached at esr@thyrsus.com.

Archive Index Issue Table of Contents

Advanced search

Understanding /dev

Preston F. Crow

Issue #50, June 1998

This article gives us a basic introduction to device files and their uses.

In designing any operating system, one question that has to be answered is, “How do programs talk to physical devices?” The UNIX answer is for the system to provide device files. When a program accesses them, they act almost exactly like ordinary files, but the kernel passes the accesses to those special files to device drivers. These files are traditionally found in the /dev directory.

To understand device files, you should start by looking at the contents of the /dev directory:

```
ls -l /dev
```

You'll see that these special files have a few differences from normal files. First, the first character of the permissions is either “b” or “c”, indicating that the device is either a block or character device. Second, instead of a file size, there are two numbers. These are the major and minor device numbers. When you access one of these files, the operating system looks up the major device number in either the block or character driver table to find the device driver (possibly with the help of **kerneld**, in the case of modules). The minor number is used by the device driver for such things as multiple instances of the same device or different aspects of the device.

You may also note a few files whose first permission character is “s”, indicating that they are sockets. Unlike devices, which are handled by the kernel, these are created by daemons when the system boots. For example, /dev/printer is a socket controlled by the **lpd** daemon. If you kill the lpd process, /dev/printer will vanish.

If you have the Linux kernel source installed, you can find out what a device is used for by looking up its device type and number in the file /usr/src/linux/

Documentation/devices.txt. With most Linux distributions, you'll find that device files exist for a number of strange operations that you wouldn't expect to use. However, you should be very careful about deleting any devices; removing the wrong files can result in making it impossible to use your system until you restore the file, after rebooting from a floppy. So make sure you have a boot floppy to rescue your system in case you make a mistake. Of course, you can't do much to mess up the /dev directory unless you are running as root.

Creating Devices

There are two ways to create device files. The recommended way is to use a program called **MAKEDEV**. You can also create them manually using the **mknod** program. We will consider them here in reverse order.

To use mknod, you need to know the name of the device, the type (character or block), and the major and minor device numbers. As an example, I once erased /dev/zero. After rebooting from a floppy and mounting my normal file system on /mnt, I wanted to create /mnt/dev/zero. I knew that /dev/zero is supposed to be a character device, with the major number 1 and the minor number 5. The command I used to create the device was:

```
mknod /mnt/dev/zero c 5 1
```

I then had to use **chmod**, **chown** and **chgrp** to set the access permissions and ownership, just as with a regular file. You'll probably want to read the man page for mknod before using it.

If that sounded like a pain, you're right. Fortunately, there's a better way. Instead of calling mknod directly, you can use a script called MAKEDEV that already knows the device types, numbers, permissions and ownership for all the standard devices. MAKEDEV uses mknod to create the devices but saves you the work of figuring out the details. For example, typing **MAKEDEV generic** will create most, if not all, of the devices you would want for a typical system. It is a good idea to read the man page for MAKEDEV before using it.

Specific Devices

/dev/null is the proverbial bit bucket. Shell scripts often redirect the output of commands to /dev/null to avoid cluttering the display with unwanted output. When used for input, it acts as an empty file.

/dev/random and /dev/urandom access the kernel's random number generator. The kernel keeps track of various “random” events, such as disk and keyboard activity, and uses a cryptographic hash to generate highly random numbers. It also keeps track of an estimate of how much random information it

has. If you use /dev/random, it will only give you as much information as it has available. If you use /dev/urandom, it will give you an unlimited amount of random information, but without as strong a guarantee on its unpredictability. Although the kernel uses this feature internally, very few programs access it through the device files.

/dev/zero generates an infinite stream of zeros. It is used every time you start up an ELF process that uses shared libraries, making it one of the most important files in /dev.

IDE hard drives and CD-ROM drives are accessed through /dev/hd*. Most newer PC motherboards include two IDE interfaces, each capable of controlling a master and a slave. hda and hdb are the master and slave drives on the first interface, and hdc and hdd are the master and slave on the second interface. The number at the end of the device name refers to the partition number, with 1 through 4 being the primary partitions and 5 and up being extended partitions.

Those familiar with other versions of UNIX will be surprised to find that all the IDE devices are block devices. Traditionally, there would also be character devices for the drives, and you would have to be careful which one you used for a given program. (For example, **fsck** wants to use the character device, but you mount the block device—not a good idea.) Linux allows programs to access a block device as if it were a character device, so there is no need for character devices for hard drives.

SCSI devices are a bit more complicated than IDE. If you have a SCSI card with multiple devices, you probably already know that each device has a SCSI ID number between 0 and 7. When the SCSI card is initialized, the operating system asks it which devices are attached. These devices are separated into several categories. The first hard drive found (the one with the lowest ID number) is sda, the first CD-ROM drive is sr0, the first tape drive is sta, and the first generic device (usually a scanner) is sga. So when accessing SCSI devices under Linux, you don't need to know the device ID number (unlike other versions of UNIX). As with IDE drives, there are no character devices for the drives, and partition numbers are appended to the drive device file.

Tape drives typically have at least two different device files. The primary device automatically rewinds the tape when you've finished reading or writing to it. The device with an “n” prepended to its name doesn't rewind the tape for you. Typically you would use this to write more than one archive on a tape.

tty devices are among the most numerous and the most confusing of all the files in /dev. Every time you log in or launch a new xterm, a tty device is

assigned to the shell. This device is responsible for displaying the shell's output and providing its input. It is known as the *controlling tty* for that shell and its child processes. The device /dev/tty is a magic device that refers to whichever tty device is the controlling tty for that process.

Besides /dev/tty, there are three types of tty devices: consoles, serial ports and pseudo devices.

Console ttys are used when the keyboard and monitor are directly connected to the system without running the X Window System. Since you can have several virtual consoles, the devices are tty0 through tty63. In theory you can have 64 virtual consoles, but most people use only a few. The device /dev/console is identical to tty0 and is needed for historical reasons. If your system lets you log in on consoles 1 through 6, then when you run X Windows System, X uses console 7, so you'll need /dev/tty1 through /dev/tty7 on your system. I recommend having files up through /dev/tty12. For more information on using virtual consoles, see the article *Keyboards, Consoles and VT Cruising* by John Fisk in the November 1996 issue of *Linux Journal*.

If you have a terminal connected to a serial port or if you dial in using a modem, you'll use a serial tty device. There are a number of different serial devices which are used if you have a special card with many serial ports (ttyC*, ttyD*, ttyX* and ttyR*), but most people just use ttyS* for the normal serial ports (COM1 through COM4 under DOS).

This begins to get confusing, because there are also *callout* devices for each serial port. For each ttyS device there is a corresponding cua device. If you're only using the serial port for one purpose, it probably doesn't matter which device you use. To be able to both dial in to your machine remotely and dial out when you're home, be sure to use the tty version of the device to listen for incoming calls and the callout version to place outgoing calls. By doing this, you won't mess up someone who has dialed in by trying to dial out on the same line. (See “Serial Port Devices”.)

Pseudo tty devices are used when a login is initiated by launching a new xterm or by connecting to your machine via TELNET. Because you can have lots of windows open and many TELNET sessions going on at one time, you may need a lot of these devices. Hence, you should have a lot of files of the form /dev/pty*. Originally, Linux limited you to 64 pseudo terminals, but some people found that too restrictive, so you can now have up to 256 of them, using different major and minor device numbers. Making it a bit more complicated, there are actually two devices for each pseudo tty: a master and a slave. Thus, for each /dev/pty* file, there's a matching /dev/tty* file. Fortunately, you don't

have to worry about the distinction other than being careful to create or delete them in pairs.

Optimizing /dev

If you wish to get every possible millisecond out of your system, there are a few things you should know about directories in UNIX file systems. (Though technically I'm talking about the implementation of the EXT2 file system, the points apply to almost all UNIX file systems.)

First note that though the command **ls** always displays the directory in a sorted order, directories are not stored with their entries sorted. Instead, each new entry is placed as close to the front of the directory as possible.

To find a file in a directory, the file name is compared with the first file in the directory. If it doesn't match, the second name is checked, and so on until either a match is found, or the end of the directory is reached. Consequently, if you have a large number of files in a directory and are frequently opening the last file in the directory, your CPU is doing a lot of comparisons. If, however, you could control the order of the entries, so as to place the most-frequently-used entries at the beginning of the directory, this would not be such an issue.

If you want to redo the internal ordering of the entries in /dev, boot from a floppy and then mount your primary file system. If your regular file system is mounted as /mnt, your regular device directory is /mnt/dev. Create a new directory called /mnt/dev2. Now you can move device files from /mnt/dev to /mnt/dev2. You will probably want to start with /mnt/dev/zero and /mnt/dev/null, as these two are opened far more frequently than any other devices.

If that sounds like a hassle, then don't bother with it. You probably won't notice any difference unless you are running on an old 386. Furthermore, the new directory cache under development in the 2.1.x kernel series will most likely make this a non-issue.

Serial Port Devices

Preston Crow grew up in Boise, Idaho. He has a Master's degree in Computer Science from Dartmouth College and hopes to soon upgrade to a Ph.D. He now works for the Open Group in Cambridge, MA, where he lives with his brilliant wife. He can be reached via e-mail at Preston.F.Crow@Dartmouth.edu.

Archive Index Issue Table of Contents

Advanced search

South African Business Uses Linux to Connect

Paul Daniels

Issue #50, June 1998

The story of a company replacing Windows systems with Linux to obtain better speed and greater reliability.

Linux is the best tool I have found for facilitating the expansion of Internet and Intranet systems throughout the company, Stocks & Stocks I.T., that employs me.

Upon arriving in South Africa from Australia, my first task was to develop and implement a system which would allow general Internet users to obtain accurate, real-time hotel room availability details from the company's chain of hotels and game resorts. This is a list of what I had to work with:

- WAN spanning South Africa
- System protected by firewall
- TCP/IP as the primary protocol
- Motorola routers

The immediate problems to deal with were:

- Each hotel/resort used Novell Netware 3.12 running only IPX/SPX.
- Each hotel/resort room database was approximately 300KB in size.
- Each hotel/resort was linked using a 64KB line (at best).
- Everyone was running around with Windows in their mind.

Getting Started

The first prototype system I set up was a central Linux web server as follows:

- Pentium 133 64MB RAM, S3-Trio 64 with 4MB RAM, ADI 17-inch monitor (Microscan), 1.2GB EIDE HDD, 8x CD-ROM, 3Com 3C509b NIC (network interface card), Ergo keyboard/mouse (Microsoft)

- Linux Slackware 1.2.13, NSCA HTTPd v1.10

The reason for having 64MB RAM, a 17-inch monitor and other “niceties” was so this system could also be used for web development of any home pages/ Intranet pages.

At each of the hotels the only system available to me was Windows 95 on which I used a scheduling system to send the room database to the central web server every morning at about 6AM. The Windows 95 machines were also being used to act as a bridge between Novell and TCP/IP. This gave the Internet user statistics on room availability that could be up to 24 hours old. The room databases were in DBase-IV format, for which I wrote a C application to search and return statistics given desired room and date details.

The system seemed fine for a while, until the Windows 95 machines mysteriously started to shut down, lock up, etc. for no apparent reason. It was also about this time that the “higher management” decided that the ability to obtain “real-time” room availability statistics was a priority.

I felt real-time statistics would be an impossibility with the current setup, as it meant sending 300KB of data to the central Linux server over a 64KB line in less time than it takes for the average person to get bored and move on to another web page.

Current Prototype

For the current prototype, the whole process was turned upside down. While still retaining the central web server, I replaced the Windows 95 machines with small, highly effective Linux machines (2.0.29). These Linux machines now process availability requests locally via a secure shell from the central web server.

In order to provide a backup, I left the existing Windows 95 system on each machine and used FIPS (First Interactive Partitioning System) to create a new partition on which I installed Linux. Details of the “remote” machines are as follows:

- Pentium 100 (Generic Clone), 8MB RAM, 800MB HDD, 3Com 3C509b NIC (no monitor, no keyboard, no mouse)
- Linux 2.0.29 (Slackware), Secure Shell (**ssh**), **ncpfs** (network file system that supports the NCP protocol used by Novell)

ncpfs is used to mount a Novell Netware volume containing the room availability database to a local directory at boot time. It should be stressed that

without ncpfs, there is no way this project would have gotten into the “real-time” phase successfully.

The process of checking room availability for hotels is now as follows (Local = central web server, Remote = Linux system at required hotel):

- Local: Get room/date details (HTML POST form).
- Local: CGI application processes POST data and executes.
- Local: Use ssh to log in to the appropriate remote Linux machine.
- Remote: Search database located on Novell Netware server.
- Remote: Exit search with 0 = Success, 1 = Failure.
- Local: ssh returns the result of the search through its exit value.
- Local: CGI application generates an HTML file with the correct response, depending on the database search result.

This whole process has been timed to take less than 10 seconds which I believe is an acceptable figure (given that there is a great deal of existing traffic on the lines). Something worth noting is that using ssh in order to remotely execute processes greatly improved operations. **ssh** offers public key encryption for all transactions and is effectively easier to setup than **rsh**.

Improving the Intranet

Now that the Hotel reservation system was up and running in a manner that pleased management, it was time to start working on the Intranet. Our primary objectives were:

- Gains in productivity
- Ease of use
- Improvements in speed

The applications the Intranet could be used to accomplish were:

- Phone/e-mail book
- Comments/Suggestions/Gripes list
- Billboard
- Training course enrollment
- Stationery/Computer shop

All the applications were written in ANSI-C. No actual HTML files exist for these applications, as the HTML is created dynamically through the application. Although this increases the time required to update the pages (having to edit, compile, etc.), it's a small tradeoff compared to the resultant simplicity. As a

minor alternative I have enabled some applications to read an external file, parse it (for any replaceable tokens) and redirect it to stdout. This too produces more overhead, but offers easier HTML page maintenance.

In order to provide database functions and non-trivial HTML POST/GET form handling, two ANSI-C object/libraries were created. All the libraries were written in ANSI C and compiled with **gcc**.

Apache and Java

After a couple of weeks, it was time to upgrade the web server to Apache 1.2b10 giving better speed and security. The upgrade to Apache from NSCA was relatively painless, with a few alterations made to the /conf/*.conf files to reflect my new directory structure.

Java was the next item on the list of “things to think about”. To me, Java is a great idea offering what C has almost been offering for a long time. However, the thought of going back from thin client into “medium-thin” was not something that interested me much. Linux is a reliable, powerful OS. I have full confidence in its ability to handle the load of the CGI applications we run. The reduction in administration by sticking with thin-client applications far outweighs the increase in server load from having to process every application transaction—this is why we chose Java.

Next Projects

Next, there was the need for an FTP-type file warehouse. Since most of the clients that are running within the Intranet are Windows-based, few are keen on using FTP. Samba was installed in order to alleviate this “barrier”. Samba enabled all Windows 3.1, 95, NT and OS/2 users to gain access to their files, applications, drivers and utilities. Of course, anonymous FTP was still set up, just in case I wasn't the only one of my kind in this company.

My most potent Linux machine is now being used to churn out images and animations using POV-Ray 3.0 which have markedly improved the appearance of the Intranet pages. I am currently working on a small (15-second) animation to be used as an “introduction” to Stocks & Stocks. To date, these are the Linux installation statistics for the company:

- Seven machines doing hotel remote access support/availability response
- Two for web application and graphic development
- Three servers

While a total of twelve machines may not seem like many for a company the size of Stocks, one must consider that there are only five NT boxes and one AS/400. (Okay, one AS/400 is often enough.)

The failure rate of the Linux systems has effectively been 0%. To date there has been only one failure, caused only by a power supply getting friendly with a bolt of lightning. Once I've installed a Linux machine, providing the user (me) doesn't do anything malicious (such as remove **agetty** or **getty**--another story), there's little cause to worry.

Some may feel Linux (and any UNIX system for that matter) has a somewhat less “user-friendly” interface when compared to Windows. However, Linux was chosen by Stocks & Stocks because it offers one thing that Windows does not—dependability. For me, Linux is “admin-friendly”.

Paul Daniels is an Australian who now resides in South Africa and works as a System Administrator for the company Stocks & Stocks. He can be reached via e-mail at jdaniels@stocks.co.za.

Archive Index Issue Table of Contents

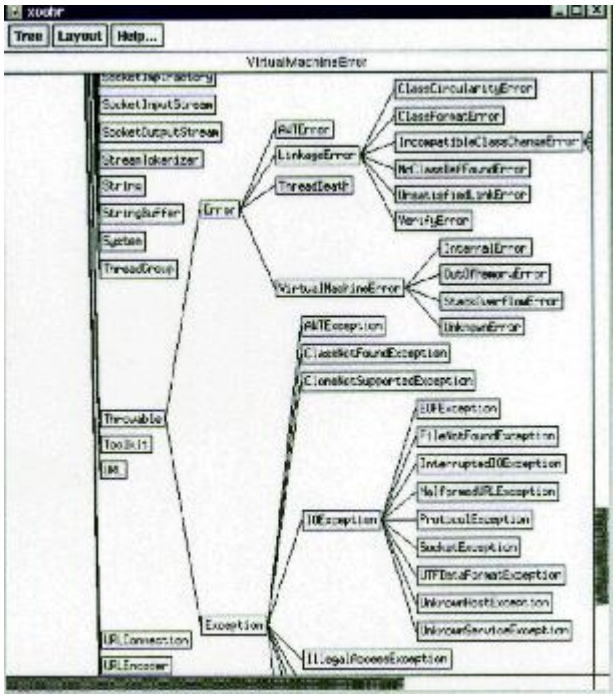
Advanced search

New Products

Amy Kukuk

Issue #50, June 1998

The OO-Browser Laptop Accelerated-X Display Server Red Hat Motif 2.1 PostShop, ScanShop and OCR Shop Qcheckbook 1.1b3



The OO-Browser

Altrasoft is shipping its newest OO-Browser, a multi-platform, object-oriented code browser which browses eight major programming languages: C, C++, Eiffel, Java, Lisp/CLOS, Objective-C, Python and Smalltalk. The OO-Browser works directly on source code, eliminating the need to assemble systems before browsing. It quickly displays and provides views of complicated inheritance trees, in order to help you understand object-oriented systems. The OO-Browser, with all languages and a full year of technical support and upgrades, is priced at $1,350 US.

Contact: Altrasoft, 4880 Stevens Creek Blvd., Suite 205, San Jose, CA 95129, Phone: 888-ALTSOFT, E-mail: info@altrasoft.com, URL: http://www.altrasoft.com/.

Laptop Accelerated-X Display Server

Xi Graphics has announced the Laptop Accelerated-X Display Server, a new version of their X server designed specifically for laptops. The Laptop Accelerated-X Display Server enables PC laptop users to utilize workstation-class features, such as multiple visuals, while maintaining high display performance. The product supports PseudoColor (16 and 256 colors), HighColor (32768 and 65536 colors) and TrueColor (16.7 million colors) modes, can handle large displays, has Hot Keys for on-the-fly switching of display resolution and is fully compatible with the native X Window System and the Common Desktop Environment (CDE). The Laptop Accelerated-X Display Server lists for $199.95 US and can be ordered directly from Xi Graphics.

Contact: Xi Graphics, 1801 Broadway, Suite 710, Denver, CO 80202, Phone: 303-298-7478, E-mail: info@xig.com, URL: http://www.xig.com/.

Red Hat Motif 2.1

Red Hat Software, Inc. has announced the availability of Red Hat Motif 2.1 for the Intel Computer. Red Hat Motif 2.1 is the full OSF/Motif development system. Among the new features of the Red Hat Motif 2.1 update are thread-safe libraries, widget printing support and internationalization enhancements for vertical text.

Contact: Red Hat Software, Inc., 4201 Research Commons, Suite 100, Research Triangle Park, NC 27709, Phone: 919-547-0012, E-mail: sales@redhat.com,URL: http://www.redhat.com/.

PostShop, ScanShop and OCR Shop

Vividata, Inc. has announced the release of Linux versions of its PostShop, ScanShop and OCR Shop software products. PostShop transforms inkjet and laser printers to PostScript-enabled ones and makes PostScript printers up to 100 times faster. ScanShop scans, prints, compresses, stores, retrieves and displays pictures and documents in full color, grayscale and bi-level (black & white). OCR Shop converts paper documents and images into editable text.

Contact: Vividata, Inc., 1250 Addison St., Suite 213A, Berkeley, CA 94702, Phone: 510-841-6400, E-mail: info@vividata.com, URL: http://www.vividata.com/.

Qcheckbook 1.1b3

Qcheckbook is a simple checkbook program built using Qddb. The motivation for Qcheckbook was to provide a tool for users to easily record checkbook transactions in order to maintain an accurate checkbook balance. Some features include memory of the last “Date” entry, sequential transactions by date, type (e.g., check, deposit, withdrawal) category and the built-in reporting capabilities of Qddb. Qcheckbook is freely available under the GNU general public license.

Contact: Herrin Software Development, Inc., 41 South Highland Ave., Prestonsburg, KY 41653, Phone: 606-886-8202, Fax: 606-277-3239, E-mail: info@hsdi.com, URL: http://www.hsdi.com/.

Help ToolKit for Motif V0.9: Binary beta Release

The Help ToolKit for Motif V0.9 has been released and is now available on the Web. The Help ToolKit for Motif allows developers to add and modify various types of on-line context-sensitive help to Motif applications. The ToolKit supports three core help types: Tips, Cues and Hints. All of these help types can be assigned to any widget and any Motif-based gadget. In addition, the ToolKit provides an API that allows a developer to plug in virtually any on-line help system, such as the Help system provided through CDE. The Help ToolKit distribution can be downloaded from http://www.softwarecomp.com/. The complete Programmer's Manual can also be downloaded in PDF format from the same site.

Contact: Software Components, Inc., 8775-M Centre Park Drive #663, Columbia, MD 21045, E-mail: info@softwarecomp.com, URL: http:// www.softwarecomp.com/.

WipeOut 1.2

Softwarebuero m&b announced the release of WipeOut 1.2. WipeOut is an integrated software development environment for C++ and Java projects. It contains a project/revision browser, text editor, class browser, make tool, symbol retriever and a debugger front end. The Standard edition is freely available. The price for the Pro edition which includes revision management and teamwork features starts at $45 for a single user.

Contact: Softwarebuero m&b, Weststr. 9, 04425 Taucha, Germany, E-mail: info@softwarebuero.de, URL: http://www.softwarebuero.de/.

Archive Index Issue Table of Contents

Advanced search

MTool: Performance Monitoring for Multi-platform Systems

Andrej Sostaric,

Milan Gabor

Andreas Gygi

Issue #50, June 1998

An article giving us all the information about a new tool for performance monitoring of distributed systems.

Quite a long title for such a small software package, but it describes exactly the content of this article. In our networked world, the amount of information is growing and becoming less manageable. The first idea for developing a performance tool came from the CERT engineer (Gygi), who works for one of Europe's academic networks—SWITCH, Swiss Academic and Research Network. SWITCH's head office is located in Zurich. Less than two dozen highly qualified engineers operate some 50 sites throughout Switzerland as well as the central SWITCH system at ETH Zurich computing center.

While SWITCH is already using some performance monitoring tools, the procedures are far from satisfactory. Each time the engineers want to monitor a remote system, they have to log in and use common tools such as **top**, **ps**, **df**, etc. This introduces undesirable security risks.

What SWITCH needed was a simple, reliable, secure tool that could be easily used from any authorized place in the network (or Internet). The tool should be compact—it must not use much of the system's resources. Furthermore, it should be multi-platform oriented. Even though SWITCH maintains mostly Sun workstations, the number of Linux boxes and other UNIX systems that they monitor is not negligible. As a first step we were interested in monitoring basic system resources: CPU/memory/swap usage and some specific processes (daemons such as **inetd**). These resources provide a snapshot of a monitored

system. In the second phase we decided to observe some additional parameters such as network traffic, disk usage, user/ftp connections and some other processes.

Figure 1. Three-Tier Architecture of MTool

Three-Tier Architecture and Java

The demands for a multi-platform, secure tool brought us to a simple decision —Java. As we have a strong students' Java team at our lab, Java was chosen as the basic development tool. Java is a network-oriented, robust tool and provides enough security for our requirements. We won't go into details of the language, but it was also clear that Java was not appropriate for the system-level programming needed to access system resources. For that reason, we chose a three-tier architecture (see Figure 1). This architecture represents a response to some of the drawbacks of the standard two-tier client/server network architecture. Client/server applications are quite easy to program; the problem is how to manage client-side software. The problem is even bigger when we require a multi-platform environment. Three-tier architecture introduces so-called middleware (e.g., the Java server in Figure 1). Basically, the middle software residing on the server represents a bridge between a client and a server. The middle tier is not only capable of handling more client connections, but also provides a more secure method of communication. Since the middleware and the client part can be written in Java, we come to a very important issue—platform independence.

Clients are no longer expected to use platform-dependent code. Even more, a copy of a Java-enabled network browser such as Netscape would suffice. Developers can now take a break, since they don't have to write a lot of platform-dependent versions of their software. This basic idea is also used on network computers (NC). Whenever the developers change the client's code, a new version can be made available for distribution on the middle-tier server. Then next time the client runs the program, the new version is automatically downloaded. The platform-dependent software which provides information to the clients over the middleware is written in ANSI C. The communication between middleware and the server is done using sockets.

Linux

As we have already mentioned, we had a daemon (**mtoold**) written for Sun workstations since they represent a majority in Switzerland's academic network. However, our environment is mainly equipped with HP workstations; we also have more than 70 Linux boxes around the classrooms and labs, as well as Windows 95/NT computers. We've learned a lot from Linux since 1993,

when we built our first Linux box using the 0.99pl15 kernel. Linux has made tremendous progress since that time.

Thus, for us, Linux as a development “playground” was a logical choice. Linux has a well documented /proc file system. (See “The /proc File System and ProcMeter” by Andrew M. Johnson, *Linux Journal*, April 1997.) Linux is the best documented operating system available. What we missed most was a good Java development tool. If Linux had a GUI like Symantec's Visual Cafe, we would have been very happy indeed. At that time, we were still forced to use the command-line Java compilers and interpreters that come with the standard Java Development Kit (JDK 1.0.2 or JDK 1.1.1). So, we used Visual Cafe under Windows for creation of the user interface, then manually added some specific code.

Architecture of MTool

Three-tier architecture turned out to be a good solution. We defined three different levels:

1. Client software
2. Item middleware
3. Platform dependent daemons

To monitor specific systems you must be connected to the network; actually, you have to connect to the middle-tier Java server. Client side presents just a GUI (Graphical User Interface). Since it is written in Java, a Java capable browser is all you need. After you make a connection to the Java server, the GUI code is transferred to your local machine. Code is then interpreted with the Java built-in interpreter (if you use Netscape, for example).

The central point of our system is the middle-tier Java server. It not only holds Java classes to be transferred to the local clients, but it also takes care of the monitoring. The Java server is a program that runs all the time but demands only a few system resources. It connects to monitored computers and asks them for new data. Communication is handled via sockets and is very simple. We also introduced a very simple protocol for data transfer.

Client Software

On the client side, one must run a starting applet to bring up the basic window in which all the computers to be monitored are listed (see Figure 2). Once we choose a computer's name from the list, a second window appears containing all the information about the observed system. We used a tabbed panel to display different types of data. We've chosen a tabbed panel to ease the

upgrade procedure. If we add new system information, there is no problem with the graphical interface—we simply add a new panel.

Figure 2. Main MTool Window

At this time we can provide the following system resources:

- CPU information (user, nice, idle, sys, average)
- Disk and memory information (free, used, swap)
- Network traffic
- Processes
- User connections

Figure 3. CPU Usage Panel

Figure 4. Disk Information and Memory Usage Panel

Figure 5. Network Transport Panel

Figure 6. Processes (preconfigured selection) Panel

Figure 7. Users Connections Panel

Middleware—Java server

The Java server is designed to run on an independent computer, but it can also run on a client of one of the monitored computers. The communication with the monitored computers via sockets is quite short and lasts only as long as the data transfer. At any time we can choose how often the Java server requests system resource data. At that time, a "ping" is sent to all monitored computers. At the ping, **mtoold** is awakened by **inetd**. Then mtoold reads system resource information and forwards that information to the middle-tier Java server. After completion the mtoold daemon "sleeps" until the next server's ping. The platform-dependent mtoold therefore uses system resources for only a very short time.

For its operation, the Java server is supported by one configuration file, named **MTool.conf**. The first part of this file contains the list of computers to be monitored, then the time between monitoring intervals, and last the list of client computers authorized to communicate with the Java server and thus monitor selected computer systems. We can use IP addresses or IP host names or both. As you can see from the sample configuration file below, one can also use the asterisk wild card in defining the domain name or the subnetwork address.

```
# First, the hosts to be moni-
# tored
cebelica.uni-mb.si
mravljica.uni-mb.si
cmrlj.uni-mb.si
strela.fcs.uni-mb.si
grom.uni-mb.si
164.8.253.16
# Second, monitoring time inter-
# val in seconds
4
# Last, clients with access to
# monitored hosts
*.hermes.si
130.216.54.51
164.8.253.99
130.15.40.*
193.246.15.*
164.8.253.*
```

Platform Dependent Daemons

The daemon mtoold is written in ANSI C and is platform-dependent. With Linux we use the **/proc** file system for CPU, memory and network information. Sun and HP do not provide such an elegant approach; on these platforms we have to use **/dev/kmem** files.

We have defined a protocol for socket communication. Information is transferred as an ASCII stream concatenated from reserved words such as **GRAPH**, **VALUE**, **disk**, **user**, **process** etc. and specific values separated with the character **|** as a delimiter. In this way, we can add a new observed parameter without difficulty. A sample stream looks like this:

```
GRAPH|LOAD_avg1|0.06|GRAPH|LOAD_avg2|0.12|GRAPH|
LOAD_avg3|0.22|GRAPH|CPU_user|1.00|GRAPH|CPU_nice|
0.00|GRAPH|CPU_system|0.00|GRAPH|CPU_idle|99.00|
VALUE|MEM_real|14652K|VALUE|MEM_free|252K|VALUE|
MEM_swap|33260K|VALUE|MEM_swap_free|31620K|
```

At this time, it is possible to intercept data transferred over the network; therefore, we are preparing Java encryption classes to enable secure data transfer. We are currently evaluating the DES and RSA algorithms. RSA would serve for key exchange (public and private) while DES, which is faster, would serve for the data transfer.

mtoold uses one simple configuration file which holds the names of the processes to be monitored. If this file is empty, information about all processes currently running on a monitored computer are transferred over the socket communication line.

At the End

Actually, at this time, we are not finished. The current version of MTool is just an intermediate step towards a more sophisticated and usable tool; however, it still provides a comfortable way of system monitoring. If we are authorized, we

can monitor selected systems from any place in the world—a Java-capable browser is the only necessity. MTool is a small, powerful tool with many benefits. While using Linux as a development environment was a good choice, we would still like to appeal to the (non)commercial software companies to provide more Java development tools under Linux. Linux and Java together represent a competitive, reliable and cheap development system.

Andrej Sostaric (andrej.sostaric@uni-mb.si) is an assistant at the Faculty of Electrical Engineering and Computer Science, Maribor and a Ph.D. student at University of Maribor, Slovenia and University of Nantes, France. His main interests are signal processing, operating systems, music, his son Miha and building a house.

Milan Gabor (milan.gabor@uni-mb.si) is a second year undergraduate student at the Faculty of EE and CS, Maribor. A great fighter for students rights and rated number one on www.experts-cxchange.com/. If there is a Bryan Adams concert near you, he'll be there.

Andreas Gygi (gygi@switch.ch) Ph.D. is system administrator and member of the CERT staff at SWITCH, Swiss Academic and Research Network, Zurich.

Archive Index Issue Table of Contents

Advanced search

An Intranet Filing System

Justin Seiferth

Issue #50, June 1998

Mr. Seiferth offers us a solution for keeping track of shared files on over an Intranet that utilizes several operating systems.

Like many of you, I often face challenges when recommending a UNIX-based solution at work. Despite its track record, many managers feel UNIX still has something to prove. So putting a UNIX system in place means working extra fast and decisively. I had just such an opening recently when I put together a system to publish the common file sharing areas of our Microsoft Windows NT-based network. I thought others might be interested in this system and created a distribution, so you too can have your own Intranet Filing System.

One of Microsoft's Windows NT failings is its archaic file system. While many of its inadequacies may be overcome, the users of our internal LAN were having problems finding information among the millions of files spread across thousands of directories located on hundreds of servers across the world. We used Linux, Sun's Solaris and some Netscape products to integrate and publish these file-sharing areas on our Intranet. You can use this system to publish any mountable file system to include FTP sites, Novell and Appleshare File Servers and NFS shares.

The key to this system working on a network utilizing Microsoft products is the ability of the Linux kernel (later 2.0 and 2.1 versions) to mount local NTFS and remote SMB volumes. There's nothing esoteric about enabling or using this option, just check it off when you're building the kernel. Once you've got an SMB-capable kernel installed, you're almost ready to go. The other components are the **smbfs** utilities, the **wu-ftpd** FTP server suite, a web server, a search engine and a javascript-capable browser. Your distribution has probably installed an operational FTP and web server, and most people nowadays have the Netscape browser installed, so all you need to do is compile the smbfs utilities and set up a search engine.

If most of the documents on the file areas you wish to publish are in text or HTML format, there are a number of search engines from which to choose: **htdig** and **glimpse**, for example. If you want to be able to search non-HTML documents then you might need one of the commercial search engines. We use Netscape's Catalog Server for Solaris, which has the ability to index binary files such as Adobe's PDF and popular office-automation application formats.

Before your “enterprise information warehouse” becomes operational, a few configuration files must be modified on your system. For instance, if you want your system to configure itself after a reboot, you'll need to modify your startup files. Also, make sure your computer is automatically mounting SMB shares into an area *accessible by an anonymous FTP session* when you boot your system and enable a few features of your FTP server.

First, let's contend with mounting shares automatically. I do this with a standard run-level 3/5 initscript; Listing 1 is an excerpt showing the most critical lines.

The **smbmount(8)** and **umount(8)** man pages have more details on what all those flags are about. Basically, we are mounting the shares into a directory accessible via anonymous FTP. Our common shares are read/write for everyone. If your site is more cautious you may want to review the implications of the file permissions and ownership or perhaps impose access controls using your file system and web server's security mechanisms.

Now, let's take a look at scripts used to configure your FTP server. If you start your FTP daemon using the **-a** option, the /etc/ftpaccess file will allow you to customize many aspects of the FTP server's performance and capabilities. Normally, you enable the **-a** option of your FTP server in your /etc/inetd.conf file. Some people run their FTP full time, in this case check the startup files in your /etc/rc.d/rc3.d or rc5.d directory and add the option there. Among the benefits of using **ftpaccess** is the ability to specify header and trailer messages for the directory listings generated by your FTP server.

A piece of our user interface is composed using an HTML file in the root directory of the FTP daemon's file area. The entry in ftpaccess is as follows:

```
...
message /welcome.html login
```

Now the contents of welcome.html are displayed at the beginning of FTP directory listings. The source of welcome.html is shown in Listing 2.

The best way to understand what this file does is to just load it up and go. In a nutshell, if the FTP listing is contained within a frame then the **if** part of the

conditional is presented. The interface allows the user to press an “upload” button which will pop open another browser instance with the FTP directory as the root window. When welcome.html is displayed as “top” within this browser; it displays instructions on how to upload a file using the FTP capabilities of the browser.

This interface is not the first one we tried. We needed a design that allowed easy navigation around a complicated system and kept at least minimal help hints in front of the users all the time. I wanted to make the system intuitive, so we could spend less time answering questions and more time working on new ideas. The tests we conducted showed most people knew how to download files but were surprised to learn they could upload or view the contents of files. We tried HTTP uploads and downloads but settled on the combination of FTP and HTTP servers distributed across several machines. Encapsulating the FTP file display simplified uploads and downloads. Unlike a web server, our FTP server labels all files as a single MIME type allowing us to use a single helper application to easily display all files. Getting the preview function to work will require the association of MIME type(s) with applications on the user's computer. We use a universal viewer. You might investigate one of the many plug-ins which allow viewing files within the browser.

Now the majority of the work and trickery is done; all that remains is the remainder of the frame-based user interface, a few snazzy graphics and some help files. When you download the distribution, you may notice that within our frame definition document we are distributing this system across several machines. This is an important feature of the system. We make use of local proxy servers for FTP and HTTP traffic to keep down the loading of our MAN/ WAN backbone. We place a proxy and web server on the department's subdomain. Since the proxy and web are local to users, we've found that lots of times the files retrieved from the central Linux-based FTP server (those ultimately stored on our SMB common file area) are served from the local proxy (cache) speeding up the file transfer dramatically and reducing our network traffic.

I mentioned before that we are using the SPARC Solaris version of the Netscape Catalog Server to allow users to expediently find any document or content within a wide variety of popular UNIX (in particular, Linux), Macintosh and Microsoft Windows application formats. We inserted a custom interface to this server which places the results into the frame normally utilized by the FTP directory display or opening splash screen. This feature provides some much needed help to users who must retrieve one of several hundred thousand documents stored on servers spread across the globe. Locating documents is absolutely not feasible using the Microsoft Windows NT file manager search feature previously recommended by the Microsoft Windows NT operators.

You can add many other enhancements such as browser access to multiple file system types (NFS, Appleshare, SMB, AFS, etc.) and Internet/Intranet FTP areas. We are also working on a configuration management add-on using PHP/FI and Postgres to present users with a fully graphical file upload facility which will also store meta data on documents such as the originator of the information, the originator's e-mail address and other information.

Resources

Credits

Justin is a Major in the US Air Force and *Linux Journal* is delivered to his office so his co-workers can also benefit from it. If you go to the Pentagon, drop by and say hello. Justin can be reached via e-mail at seifertj@www.disa.mil.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #50, June 1998

Our experts answer your technical questions.

Multithread Applications

Which library do I need to use for multithread applications? I found pthread.h in /usr/include/, but I didn't find libthread.a in /usr/lib/ as I can in AIX. —Ju Rao Slackware

There is an excellent threads library available called LinuxThreads. You can download the latest version from ftp://ftp.inria.fr/INRIA/Projects/cristal/Xavier.Leroy/. [For more information on LinuxThreads, see http://pauillac.inria.fr/~xleroy/linuxthreads/.] —Chad Robinson chadr@brt.com

Missing svgalib

Hi, I am a new user, and I have what I am sure is a simple question. I upgraded the kernel from version 1.2. Now when I run games, I get an error about a missing svgalib. Consequently, I cannot get any picture. I am using an ATI 3D Expression with 2MB. How do I edit the config or lib file in order to indicate which libraries I have? —Jonathan Barrie Slackware

I'll bet you upgraded the libraries as well as the kernel. Nowadays, everything uses shared libraries, and programs can't run unless the right shared libraries are installed.

I'd try **ldd <program>** to check which libraries it finds and which it doesn't. Once you know which libraries are missing, you can then install the Slackware packages that provide them.

You can also run **ldconfig -v** to see which libraries are installed; if your program looks for a different version of a library already installed, look for “backward compatibility” packages.

Note that newer distributions trace dependencies between packets, so you are unlikely to experience this error when using them. —Alessandro Rubini rubini@linux.it

Customizing Login Screens

How do I customize the “Login” screens for TELNET and FTP logins? —Robert Farrell

For TELNET, create and edit the /etc/issue.net file with whatever you wish to be displayed for the login screen. You can also edit the /etc/motd file to display the message of the day after someone logs in.

For FTP, go to the root directory of the FTP space (usually /home/ftp/). You can verify the location and file names by looking in /etc/ftpaccess. Edit the welcome.msg file to whatever you wish to be displayed. If you want specific messages to be displayed when users enter into a particular directory, edit the .message file in that directory. —Mark Bishop mark@tct-net.org

Backup Scheme

In a 50/50-mixed NT/Linux LAN of four to six servers, what is the best tape backup scheme you recommend? Where should I install the backup units—on the NT or Linux or both? What tape backup utilities should I use? What if Solaris 2.5 servers are added? —Jon ChunRed Hat

tar has always been the standard backup tool in the UNIX environment. If you want something more powerful, you should investigate **cpio** or **afio**, both of which are standard, command-line tools. BRU and other commercial backup facilities exist as well.

If you want to back up mixed NT and UNIX systems, you will most likely need to use a dedicated backup facility for the NT servers. A backup of NT is nice but backing up your domain server won't be very useful unless you include the registry settings, which only an NT backup tool can do.

The best possible solution is to install a tape drive in each server and use that server's own tools (NTBackup on NT, tar on UNIX) to back it up. That method will provide you with the fastest recovery times as you won't be restoring across the LAN.

If you don't need to cut costs, you can investigate ARCServe or another commercial backup utility. These tools include clients for many platforms, and you can probably get the SCO client working under Linux using iBCS2. That

would allow you to use a Windows NT machine to back up your UNIX servers. —Chad Robinson chadr@brt.com

You can use the SMBFS functionality of the Linux kernel in order to mount remote NT servers on your Linux system:

```
smbmount //NT_SERVER//FOO /nt_server
```

Then you can save these directories with tar or any other backup utility.

A more efficient solution (but not free) is to use commercial software such as ARKEIA or BRU which support remote NT/95 system backups. —Pierre Ficheux, Lectra Systèmes pierre@lectra.com

Using a Teles Card

I have a Teles ISDN card, which I would like to use under Linux. After looking around a bit I found a driver in the Red Hat distribution. So, I compiled a new kernel with support for the Teles card. However, I have no idea how to use it. With a standard modem I can talk with **at** commands to /dev/modem/, but how does it work with the ISDN card? —Rien Broekstra Red Hat

You need the isdn4k-utils. There are other packages, e.g., vbox for building an answering machine, but this is basic. Check ftp://ftp.franken.de/ or read /usr/src/linux/Documentation/isdn/* for locations of additional packages. —Ralf W. Stephan stephan@tmt.de

I'm fairly certain that you didn't need to recompile your kernel. The Red Hat kernel packages build ISDN support as modules. (I'm sure they do as of 5.0, and I think we did in 4.2 as well.)

Anyway, the best place to find this type of information is in /usr/src/linux/Documentation/isdn/README.HiSax/. Everything you need should be covered there. —Donnie Barnes redhat@redhat.com

Mysterious Temporary Files

I have 160 users in my system. They all use this system strictly as an e-mail server with Netscape as the client. I have noticed that my free disk space has been shrinking over the year. Looking around, I found a whole boatload of files in the /tmp directory which appear to be mail messages with graphic attachments. The file names are pop3a0XXXX and many of them are the same size, have the same owner and have creation dates 2-5 minutes apart. How do I fix this? Does the in.pop3d have a problem? —Mike Gasiorowski mgaz@blue.friendswood.isd.tenet.edu Slackware

It looks like your daemon is losing its temporary files; this can happen when users hang up without closing the connection. (Even though the user is causing it to happen, it should still be considered a bug.)

The easiest solution to the problem is to use **cron** to do a periodic check of /tmp with the following command:

```
find /tmp -daystart -atime 1 -exec rm \{} \;
```

--Alessandro Rubini rubini@linux.it

Mounting a DOS Partition in Linux

Can I mount a DOS logical partition in Linux? I am trying to mount my DOS D: and E: drives. They are both logical partitions within an extended partition of my DOS system. Is this possible in Linux? If so, how do I mount them? Thanks for your help. —Andrew Hamlin Slackware

Yes, this is possible. When Linux boots, you receive a message describing the partitions found on each drive. You can review these messages after Linux boots using the **dmesg** utility.

Primary partitions are numbered from 1-4, for example, **hda3**. Extended partitions will be numbered from 5 and up, so the first extended partition on drive **hda** will be **hda5**. Add an entry for this drive to your /etc/fstab file, and you will be able to mount it. You can test this manually by typing:

```
mount /dev/hda5 /mnt -t msdos
--
Chad Robinson
chadr@brt.com
```

Installing on a ESDI Drive

How can I install Linux on an ESDI drive? The machine is an IBM portable. The drive is not recognized by the following boot disks: bare, SCSI and idecd. —Chriptopher Ochal

If the hardware is Micro Channel-based, you'll need to get a modified boot disk from the Micro Channel Linux Home Page (http://www.glycerine.itsmm.uni.edu/mca/). Instructions are located at http://glycerine.itsmm.uni.edu/mca/general-goods.html#Slackware/. —Steven Pritchard steve@silug.org

top

I started with Slackware 1.0.2, and **top** worked fine. I recompiled the kernel for some reason, and top no longer worked. I installed 2.0.0 when it was released and have patched it up to 2.0.31. **top** still doesn't work. It thinks for a moment then blows out, usually taking the **xterm** or **rxvt** with it. I have downloaded all the supporting stuff listed as required for 2.0, and I have installed more memory. **free** and **ps** work fine. I do not have a clue where to look for top information. Man pages don't have much. Help! —Bill

Make sure you are running the latest version of **procps**, available from ftp://sunsite.unc.edu/pub/Linux/system/status/ps/

Occasionally the layout of proc files changes, which breaks older versions of the **ps** utilities, including top.

You might also wish to make sure your **termcap**, **curses** and **ncurses** are all up-to-date. —Steven Pritchard steve@silug.org

Archive Index Issue Table of Contents

Advanced search