

Advanced search

Linux Journal Issue #84/April 2001



Features

Focus: Internet/Intranet by *Don Marti*

Providing E-mail Services for a Small Office by *Stew Benedict*

The ideal, simple small-office e-mail solution.

Freenet Installation and Administration by *Peter Todd*

Get involved in the most exciting file-sharing technology.

oftpd: A Secure, Modern FTP Daemon by *Don Marti*

oftpd's simplicity gives it a performance edge.

Linux on Carrier Grade Web Servers by *Ibrahim Haddad and Makan Pourzandi*

A great software solution for web traffic problems.

Indepth

Managing Initscripts with Red Hat's chkconfig by *Jimmy Ball*

Control your services with the chkconfig utility.

Using Mix-ins with Python by *Chuck Esterbrook*

Python provides an ideal language for mix-in development.

Managing Your Money with GnuCash by *Robert Merkel*

A tutorial on a powerful, free accounting program.

Toolbox

Linux Means Business Enterprise-Level Health-Care Applications
by *Gary Bennett*

At the Forge Server-Side Java with Jakarta-Tomcat by *Reuven M. Lerner*

Cooking with Linux [Managing Multiple Cooks](#) by Marcel Gagné
Paranoid Penguin [Battening Down the Hatches with Bastille](#) by
Mick Bauer
[GFX XFree86 and Video4Linux](#) by Robin Rowe

Columns

[Linley on Linux Turbulent Start for Transmeta](#) by Linley Gwennap
[Focus on Software](#) by David A. Bandel
Focus on Embedded Systems [Free Beer vs. Free Speech](#) by Rick
Lehrbaum
Linux for Suits [The New Vernacular](#) by Doc Searls
[Games Penguins Play Descent3 for Linux](#) by Neil Doane
[.org Watch LinuxPPC Goes Nonprofit](#) Leslie Proctor

Reviews

[Appgen Moneydance 3.0](#) by Joseph Cheek
[CorelDRAW Graphics Suite](#) by Choong Ng
[Firebox II](#) by Glenn Stone

Departments

Letters

upFRONT

From the Editor [Lunacy Floats](#) by Doc Searls

[Best of Technical Support](#)

[New Products](#)

[Archive Index](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Focus: Internet/Intranet

Don Marti

Issue #84, April 2001

Three key problems face Linux as an internet platform today: security, proprietary software and the high cost of popularity. Naturally, *Linux Journal* will help you attack all three.

Linux comes from the Internet and thrives on the Internet. As the desktop Linux community struggles with proprietary formats and user lock-in, you might think those of us who are just doing network services have it easy. Wrong. Three key problems face Linux as an Internet platform today: security, proprietary software and the high cost of popularity. Naturally, *Linux Journal* will help you attack all three.

Networks need high-quality, secure software. Open-source licenses and development models are an important step toward that, but last year *Linux Journal* warned the shovelware mongers, I mean Linux distributions, that their default installs were loading users up with too many potentially exploitable services and that they should start locking things down by default. But did they listen? No. Now we have the "Red Hat Ramen Worm", and proprietary software PR people blaming Linux for one distribution's irresponsible decision. From what I see, though, it could have been any of the distributions, so if you're not Red Hat don't think I'm not talking to you too.

Mick Bauer is writing about Bastille this month, and it certainly helps, but the very existence of a security package as an add-on is profoundly backward—like taking delivery of a car with no bumpers or seatbelts and having to get a local mechanic to install them. Exploits happen, but worm epidemics wouldn't be a public embarrassment for Linux if the distributions would just make a secure posture the default.

If you must run FTP (and most systems don't need to) please, please don't run some huge, deluxe, featureful FTP daemon written back in ancient times, when there was no secure alternative to FTP for password-protected file transfers.

Use a minimal “anonymous-only” daemon such as **oftpd** (page 92) and be happy.

It might be hard for some of you to imagine the prospect of not having e-mail at work—but that's where Stew Benedict found himself. It's an old story for us but an inspiring one—set up a single, inexpensive Linux box to offer e-mail to lots of users. Internet e-mail dramatically changes work environments, as lots more people get information and help from the outside. Especially in bleak, brain-stifling places it changes the way people work for the better—and Stew did it on the smallest of budgets. Read how on page 100.

No networking issue would be complete without some discussion of the question “How can I get more performance out of my web site?” Ibrahim Haddad and Makan Pourzandi find an answer from the bottom up, using the classic “load balanced cluster” approach and the free Linux Virtual Server Project software [see page 84]. That's a workable solution for your business web site today—don't let yourself get bamboozled into proprietary load balancing or clustering.

When Richard Stallman wrote “Join us now and share the software, you'll be free, hackers, you'll be free,” he forgot “sane”. License managers, incompatible binary-only kernel modules and lack of tweakability are sending more than one webmaster over the edge. So get the free stuff and have time to make the site better, don't just work around some idiot marketing person's conception of how your site should work.

The Web is great, but the tragedy of popular independent web sites is that they start to require more bandwidth and server power than their creator can afford. At that point, the webmaster either “sells out” to a business that starts carrying ads, tracking users and doing other nasty stuff, or the site dies out. Naturally, the vendors of big bandwidth and big iron love this.

Freenet to the rescue. This much-hyped system really works—Peter Todd sent us his Freenet article over Freenet. Think of it as a distributed Berkeley DB, but one where you can get your data back from any Freenet “node”, not just the one where you stored it. It's a little more complicated than that, but honestly, not much.

Freenet's decentralized architecture was originally intended to keep the Man from suppressing underground newspapers and hash brownie recipes. But more importantly, it's becoming an architecture for sustaining web-like sites and Usenet-like groups while sharing the costs of servers and bandwidth among all the readers, not just slamming the creators. Get a DSL line, join

Freenet (page 96), and soon your favorite sites won't have to make the decision of whether to shut down or sell out.

The fundamental advantage of packet-switched networks is that they disproportionately reward economically efficient and moral behavior. But like the HVAC system in *Brazil*, networks “don't fix themselves, sir.” Put some effort into security, free software and free peer-to-peer systems, and the network will pay you back many times over.

Peace and Linux.

—Don Marti, Technical Editor

Resources



[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Providing E-mail Services for a Small Office

Stew Benedict

Issue #84, April 2001

If your company isn't ready to dive head first into DSL or full-service Internet and e-mail service, a system like this might be a good solution.

The company I work for is what could euphemistically be called thrifty. We've got equipment dated from the 1930s, and I worked there three months before I found out we even had an internet account. (Pretty bad for the system administrator, huh?) This was my first true IS job, and I was willing to deal with the situation to get the experience. Come to find out, the account was being hoarded in the engineering department by the owner's son and his buddies.

As time went on, our customers began requiring us to be more in step with the times, accessing information on the Web, sending and receiving e-mail, etc. I saw a number of other ways that adding a Linux box to the network could help out the company and pay for itself in no time. So I wrote a proposal and got permission to procure a box to act as a file server, fax server, internal web server, e-mail server and Internet gateway. Aside from the guys in engineering, the rest of the plant had not even used e-mail, and engineering had only used it once or twice. They thought the Web was the only thing to do on the Internet. We won't talk about what sites they were visiting.

The focus of this article is on the e-mail portion. I intend to cover how to use sendmail, fetchmail and procmail to allow a small company to effectively "hide" behind a single e-mail address. Now, before you run out and do this, you might want to consider how your ISP may feel about it. In our case, the volume of e-mail is small, maybe 20 messages a day. Compared to my personal account, which gets 100-400 a day, plus my business account, which gets maybe 20-30, this is peanuts, and I don't think our ISP has much to complain about. Your mileage may vary.

Did you ever notice how return addresses look when reading e-mail? Most times, if the person has their full name set up in their e-mail client configuration or if the administrator has it in `/etc/passwd`, you will see something like:

```
Firstname Lastname <somehandle@someisp.com>
```

I was convinced that I could use this, somehow, to rewrite the outgoing return address in such a way that, when the mail was answered, I could pass it through a filter and on to the appropriate person, while only consuming one e-mail address from our ISP. I ended up buying O'Reilly's *sendmail* book and doing a couple weeks of experimenting, but I've now got a solution that has been in place for about two and a half years and works pretty well. At an additional \$5 per e-mail address per month from our ISP, and about 15 users, I think it was worth the trouble.

Here's the plan. The names are, of course, fictitious.

- `thriftcompany@someisp.com`—my company's e-mail address from our ISP.
- `smtp.someisp.com`—our ISP's SMTP server (outgoing mail from us).
- `pop3.someisp.com`—our ISP's POP3 server (incoming mail to us).
- `linuxserver.thriftcompany.com`—our file server (`thriftcompany.com` is strictly internal, not a registered domain).
- `mrpserver.thriftcompany.com`—a Sun box, hosting our MRP system. This is not a necessary part of the plan, but many of my users spend all their time on an xterm to this box, so I installed Pine on the Sun box for them. (Pine, in the GNU tradition, stands for Pine Is Not Elm—another text-based e-mail program. Pine is actually an extremely efficient mail client, and my program of choice. It would take me days to wade through my e-mail with a GUI.)
- `thriftcompany@linuxserver.thriftcompany.com`—bogus default e-mail address on the Linux box. All the incoming mail goes through this address, and procmail passes it on to the users.

Through some `/etc/sendmail.cf` magic, and a procmail filter on the incoming side, I can let several users use the same e-mail account yet, for the most part, keep their mail private.

Outgoing scenario: 1) User writes e-mail, either in Pine on the Sun, or Outlook or Netscape's mail client. I only set up GUI accounts for folks who need to deal with attachments. 2) If mail is written on the PC, it is configured to pass the mail up to the Sun box and retrieve mail from the same location. 3) If it's an internal address, the Sun box delivers it. 4) If it's external, it is passed to the Linux box. 5) Outgoing mail on the Linux box is queued and sent out in batches twice an

hour, after incoming mail is pulled in. 6) At the proper time, `/usr/sbin/sendmail -q -v` is run and, through the use of `/etc/genericstable.db` and some masquerading rules in `/etc/sendmail.cf`, the user's return address is rewritten as **Firstname Lastname <thriftycompany@someisp.com>**

Incoming scenario: 1) Server makes connection to ISP at time defined in cron. Or if already connected, leave the connection alone. This is handled by **pppd** and **diald**, but that's another article. 2) Incoming mail is retrieved using `fetchmail` and passed to the default account, `thriftycompany@linuxserver.thriftycompany.com`. 3) Procmail filter in `thriftycompany`'s account looks for proper names in the incoming addresses and passes the mail on to the users. Mail that does not fit a filter rule falls in the default mailbox, and as a precaution, a folder for each user is set up in this account's mail/ directory. 4) All local mail, except root and `thriftycompany`, gets forwarded to the Sun box. 5) User either receives mail in real time in Pine on the Sun box, or interactively retrieves it from the Sun box using GUI client. 6) All this sounds complicated, but it's really not. Remember, the Sun box is just a convenience for my situation and not necessary to the process. You could merely have your users grab their mail from the Linux box.

Now for the files. Most, if not all, of what you need should be in most Linux distributions already or somewhere on your CDs. Just in case you need assistance, you will find URLs in our Resources section.

Local Mail Handling

Let's get the Sun/Linux mail passing out of the way first, so if you don't need the second server, you can ignore it and move on.

The Sun box's `/etc/mail/sendmail.cf` is standard. When I set this up, I was still new to Solaris and hesitant to fool too much with a system running our whole plant, so I did this to pass mail out `/etc/mail/sendmail.cf` entry:

```
DRmailhost
CRmailhost
```

`/etc/hosts` entry:

```
192.9.200.2    linuxserver mailhost
```

Before I made the change, `/etc/hosts` had the mailhost entry after the Sun's IP address:

```
192.9.200.1 mrpserver mailhost
```

This seemed to do the trick to get outgoing mail over to the Linux box.

On the Linux box, we want all local mail to go to the Sun box, where the users' mail folders reside. None of my users, aside from me, directly log in to the Linux box. They use files shared through Samba. Telnet and FTP access is closed off:

```
DHrelay:mrpserver.thriftycompany.com
```

The exceptions to this are root and the thriftycompany user, whom I'd like to stay on this box:

```
CL root thriftycompany
```

That about covers the interaction between the two local UNIX boxes. Local mail stays on the Sun, internet mail gets passed to Linux and then queued for the next connection. Incoming internet mail will get re-addressed to local users and then relayed to the Sun box. The root and thriftycompany accounts on the Linux box stay put, and I check those as part of my daily routine.

Internet Mail, Outgoing

Some of this setup was taken from various HOWTOs over the years; other parts I gleaned from scouring Usenet and the O'Reilly *sendmail* book.

sendmail startup: for a demand dial scenario, we don't want sendmail initiating the connection each time there is mail in the queue, so you need to edit your sendmail startup line to hold off on "expensive" mail:

```
old entry: /usr/sbin/sendmail -bd -q15m
new entry: /usr/sbin/sendmail -bd -os
```

On a Red Hat-based system, this would be set up in `/etc/sysconfig/sendmail` and/or `/etc/rc.d/init.d/sendmail`.

You'll be defining which mail is expensive within the `/etc/sendmail.cf` file in Listing 1, as well as telling sendmail to hold this type of mail.

Listing 1. Defining and Holding Expensive Mail

Notice the "e" in the "F=" portion for smtp, esmtp and smtp8. This is the "expensive" flag, and we leave it off the local relay. Also the Mlocal and Mprog should not have this flag so that local system mail gets delivered immediately.

A cron job connects to the Internet twice an hour, and as part of that job, we will send out all the queued mail once the internet connection is in place:

```
/usr/sbin/sendmail -q -v
```

Now to get the outgoing mail delivered and not rejected by domains on the Internet. Since we do not have a valid domain name, we need to do some work on the return address. We need to masquerade the return address, as well as the envelope, and to get any replies back to the original sender, we need to rewrite the "From:" address.

Masquerading

If you are building up your sendmail.cf from m4 sources, then your local .mc file needs to contain the following:

```
MASQUERADE_AS(someisp.com)
FEATURE(masquerade_envelope)
FEATURE(limited_masquerade)
FEATURE(genericstable)
```

We are using limited masquerading so only hosts defined in CM get masquerade. If you are just editing /etc/sendmail.cf, then the following lines need to be modified as shown:

```
# who I masquerade as (null for no masquerading)
  (see also $=M)
DMsomeisp.com
```

You will probably also want to relay the mail through your ISP so that any downstream mail servers see the mail as coming from a valid domain:

```
# "Smart" relay host (may be null)
DSsmtp:smtp.someisp.com
```

Define the domain names that should be converted to the masqueraded address:

```
CG mrpserver.thriftycompany.com
CM mrpserver.thriftycompany.com
```

(If you've got just the one box, this would be linuxserver.thriftycompany.com.)

Now, the sendmail.cf lines to masquerade the contents and the envelope get a bit messy. You would probably be better off building a sendmail.cf from the m4 sources as shown in Listing 2.

Listing 2. Building sendmail from M4 Sources

The final piece of the outgoing puzzle is to get the user's return address rewritten. If I were to compose a message on the Sun box, Pine would put together a return address that looks something like this:

```
Stew Benedict <stew@mrpserver.thriftycompany.com>
```

Looks good enough, except that it is not a real address out on the Internet, and I would never get a reply to my message. Plus, most mail systems would reject it coming in as a nonexistent domain address.

What I want it to look like is this:

```
Stew Benedict <thriftycompany@someisp.com>
```

This is where sendmail's "genericstable" feature will finish up the job. Again, if you are building up your sendmail.cf from m4 sources, the following line will do the trick:

```
FEATURE(genericstable)
```

In the sendmail.cf file, if you are hand editing it:

```
# Generics table (mapping outgoing addresses)
Kgenerics hash -o /etc/genericstable
```

(The -o means "optional", so sendmail will not halt on startup for lack of the file.) This addition to your local .mc file generates the block shown in Listing 3 in sendmail.cf.

Listing 3. sendmail.cf

The genericstable.db file is built up from a text file of the following format:

```
stew      thriftycompany@someisp.com
joe      thriftycompany@someisp.com
```

This file is then fed to the **makemap** program to create the db file:

```
makemap hash genericstable.db < genericstable
```

That does it for outgoing mail. Once you have finished creating/modifying sendmail.cf and creating the genericstable.db file, you will need to restart sendmail. On a Red Hat-based system, this is done with:

```
/etc/rc.d/init.d/sendmail restart
```

Internet Mail, Incoming

For incoming mail we use **fetchmail** to fetch the mail from the ISP. My cron job makes the internet connection and then runs a line like this:

```
su -c "/usr/bin/fetchmail -a -f /home/thriftycompany"

thriftycompany's .fetchmailrc:
poll pop3.someisp.com proto pop3 user thriftycompany
```

Procmail is defined as the local MDA (mail delivery agent) in the sendmail.cf file:

```
Mlocal,      P=/usr/bin/procmail, F=lsDFMAw5:/|@qShP,  
            T=DNS/RFC822/X-Unix,  
            A=procmail -a $h -d $u
```

All the incoming mail goes to the thriftycompany account, where there is a .procmailrc file set up to parse the incoming "To:" lines and forward to the appropriate users, as seen in Listing 4.

Listing 4. Forwarding E-Mail to the Correct Users

You can also use this procmail filter to filter out ILOVEYOU-type viruses, limit or quarantine attachments and other useful things. Check out the procmail docs for more info on this. Each user's mail is held in a folder under thriftycompany, in the event of accidental erasures, etc. I periodically purge these folders by hand.

Logging

My boss has not quite bought into the usefulness of the Internet and requests full tracking of its use. Part of my tracking includes logging of all the incoming and outgoing mail messages—quantity and size—per user. This is done through a shell script run by cron every morning, as shown in Listing 5. The output from this job looks like Listing 6.

Listing 5. Tracking Incoming and Outgoing Mail

Listing 6. Output of Morning Cron Track

That about does it. The tricky part is getting outside folks to address the incoming mail properly. For most mail clients, this just requires making a First Name, Last Name and e-mail address entry in the address book, with the person's proper name and our ISP e-mail address. For those people who have regular correspondents who just can't seem to get it right, I add a procmail rule with the "From:" address to make sure the mail gets to its proper destination. The other suggestion I give users is to send an e-mail to the other party and let them add the return address from that to their address book.

Maintenance

There is a certain amount of maintenance to this system on my part, but it's minimal. In writing this article, I've considered ways to automate these portions too; but for now, it's not really much of a burden.

I check thriftycompany's mail daily to check for messages that fell through the procmail filter. Not a big thing, I could forward this mail to myself, and then I'd

see it sooner, or I could set up **KBiff** (KDE mail notification utility) to watch this mailbox.

Also, I like to purge the individual mail folders under thriftycompany's e-mail account. Again, not a big thing. I'm not hurting for space on this server and our volume is small. There's probably a Perl script out there that I could use to prune messages "n" number of days old.

Currently, when I add a new user, I have to add a new entry to the genericstable file, run makemap, restart sendmail and add an entry to the procmail filter for thriftycompany. I've considered making a shell script to accomplish these steps too, but currently I add a user maybe four times a year.

I hope this article has given you some insight into setting up an e-mail solution for a small company, one not quite ready to make the leap into a full DSL or T1 connection with a domain name. If you need to use the Internet and e-mail to communicate with your customers and vendors, this should give you what you need to get the job done.

Resources

Stew Benedict is a Systems Administrator for an automotive manufacturer in Cleveland, Ohio. He also is a freelance consultant, running AYS Enterprises, specializing in printed circuit design, database solutions and utilizing Linux as a low-cost alternative to commercial operating systems and software. He has been using and promoting Linux since 1994. When not basking in the glow of a CRT, Stew enjoys time with his wife, daughter and two dogs at his future (not too much longer!) retirement home overlooking Norris Lake in the foothills of the Smokies in Tennessee.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Freenet Installation and Administration

Peter Todd

Issue #84, April 2001

Peter demonstrates how to take advantage of the World Wide Web alternative, Freenet.

Freenet, quite possibly the most exciting file-sharing technology out there, is unfortunately a little difficult to install and administer. Your intrepid author spent a weekend with literally no sleep (strong tea and death metal can do wonders) figuring it out when he first got his hands on it back in September 2000. Fortunately, Freenet's ease of use has come a long way since then. With an hour or two of work, anyone competent in Linux system administration should be able to get their own Freenet node up and running.

The installation of a Java runtime is out of the scope of this article. However, a list of recommended JDKs (Java Development Kit) and some warnings about them can be found in the Resources section.

If you want to run a full-time, contributing node you'll need 100MHz 486 or greater with at least 32MB of RAM. There are no minimum requirements (other than about 1MB for the actual software) for hard drive space, but in this age of 20GB drives, it'd be nice if you set aside a few GBs for Freenet data. For your internet connection, a static IP address or dynamic DNS is required. Anything faster than a dial-up is fine. As for uptime, if your node is on 24/7 you can contribute (downtimes of up to an hour or so are okay, but try to avoid anything more than that). Anything less and your node will still work just fine for you, but you won't be contributing anything to the network.

For a part-time, noncontributing node, similar system and memory requirements apply, but you can use whatever internet connection you like.

Basic Installation

First of all, you need to actually download Freenet. Here you have two main options. You can download a binary or choose the source package. If you run Debian, Mr. Bad has created a Debian package (see Resources). Because Freenet is written in Java, you don't have to worry about binary compatibility, so it's easiest simply to download a binary package. After that the question is which version do you want. A release (current is 0.3.5) or CVS copy? You'll probably want to go with a release at first. Do make sure you download the Linux tarball, not the Windows executable.

The Linux version of Freenet doesn't have any installation scripts. Instead you simply untar the tarball into a directory of your choice. Personally I'd suggest creating a special Freenet user and putting the Freenet software and data store in the Freenet user's home directory.

Next you'll want to make your node automatically start at boot up. Unfortunately, not all distributions handle the startup scripts quite the same way. If you want to use `dæmontools`, follow the instructions at freenet.netunify.com/25. Otherwise, use something like:

```
su - -c "cd ~/freenet ; rm freenet.log nohup.out ; nohup ./freenet_server & >/dev/null" freenet
```

This changes the user to the Freenet user. The `"-"` sets the environment to that of a login shell so the environment is set up as though running from a normal shell. Finally, the command `-c` changes the directory to the Freenet directory, removes old log files and runs the Freenet server in `nohup` mode directing any non-errors to `/dev/null`. You'll need to insure the Java program itself is in your path. Try running Java from a root shell to test this.

Configuration

The Freenet node software uses two configuration files: `.freenetrc`, the configuration for the node; and `.fproxyc`, the configuration for the `fproxy` module. Of the two, `.freenetrc` contains all but one of the options; `.fproxyc` just tells `fproxy` which Freenet node it should connect to.

The first thing you'll want to set is the `transient` option up at the top of the `.freenetrc` file. The default, `"no"`, means that your node will tell other nodes about its existence. This is probably what you want if you plan on running a 24/7 node with a good internet connection and a static IP address or dynamic DNS service. If you set `transient` to `"yes"`, your node won't tell other nodes about its existence. If you have spotty uptime, a slow internet connection or a dynamic IP address without dynamic DNS, set `transient` to `"yes"`.

You'll want to change the port used by Freenet to a random port between 5,000 and 65,535. If everyone ran their node on the same port it would be far easier for Freenet to be filtered out, which is not a good thing. To do this, change the listenPort value in .freenetrc and the serverAddress in .fproxyc to reflect the new port. Also, remember to use the -serverAddress option when running the command-line clients or set the FREENET environment variable to the address of your node.

If you have a dynamic IP address but use dynamic DNS you can still contribute to Freenet by setting nodeAddress to your dynamic DNS name. Instead of telling other Freenet nodes the IP address of your machine, it will tell them the address in the nodeAddress setting.

Resource Configuration

By default, Freenet will store a maximum of 1,000 items in the data store and use a maximum of 500MB of hard drive space. The diskCache and dataStoreSize options control this. Note that, due to bugs, the actual amount of space used can easily go over this. Often files in the data store simply get lost and don't get deleted when they should be. Restarting the node periodically, every day or week, will clean out those lost files. Secondly, while transferring large files the node will store the whole file while it's in transit even if doing so makes the node go over the size limit. Hopefully by the time you read this article, these bugs will have been fixed, though the second problem is difficult to fix for various reasons.

After that we have the dataPath and dataPropertiesPath that control where the data store is (the default is the .freenet directory where you installed Freenet). The difference between the two is that dataPath is where the actual data gets put, and dataPropertiesPath controls is where the .inf files describing the data stored get put. In all but unusual cases the two will be the same value. You may want to set this path to another partition with more space available than the default. Any filesystem that supports long filenames is fine. Note that you can't use multiple partitions simultaneously, you'll have to run multiple nodes on different ports for that.

There is also an option to limit bandwidth used, bandwidthLimit. This limit applies to outgoing data only. Incoming data is not limited (see maximumConnectionThreads). For most people the 50K/sec limit is probably fine, but if you can, please increase it or disable the bandwidthLimit to about a third to a half of your outgoing bandwidth, about 50K/sec to 300K/sec for your average cable or DSL line. Note that the bandwidth limit does not apply to requests originating from the same machine.

Incoming bandwidth can be limited, to a degree, with the `maximumConnectionThreads` option. This option limits the maximum number of incoming connections from other nodes. Changing this changes the amount of bandwidth and memory used. Most people can probably leave it at its default of 50K/sec, but if you have memory restrictions (or excesses) you can reduce or increase it.

Note that there is currently no way to limit the amount of total data transfer in a given time period. So as of yet, there is no way to only allow, for example, 1GB of data transfer per month.

Other Options

At the end of the configuration file are the options that control logging. A single log file is used, `freenet.log`, by default (controlled by `logFile`). The logging option controls the threshold of events logged. The default, "normal", doesn't log any of the mundane details of your node sending and receiving data. The "minor" setting can often provide useful debugging information, however debugging is overkill. Setting logging to minor is probably your best bet.

Finally, there is one last option that's not in the default configuration file, `informDelay`. Before writing the node's address to `informURL`, the node will wait a default of 24 hours to make sure your node is stable. Your node must be running for 24 continuous hours before the `informURL` is notified. Initially, one of the big problems with the `informURL` system was that people would try Freenet for a few minutes, decide it wasn't worth it and remove the Freenet software. Meanwhile, the address of their nonfunctioning node was sent to `informURL`. When nodes tried to get a working address, almost every single address in `informURL` would be nonfunctioning. It got to the point where one of the big problems in getting a node up and running was finding another node!

If you plan on restarting your node once a day you'll have to change the `informDelay` to something different. Setting it to 0 by adding the line **`informDelay = 0`** to the configuration file will disable the feature and is what you want. Otherwise don't touch `informDelay`, please!

Testing

Now that you've got your node up and running you want to see if it works, right? First try running:

```
freenet_request test-key
```

If that works you'll see the message: **Congratulations! You've fetched a Freenet key** on the console after some debugging information. If it doesn't do anything for more than two minutes or so it's not working.

Next try the URL `http://localhost:8081/KSK@Aardvark` in your favorite web browser to see if `fproxy` is working. If it is, you'll see the page of Aardvark's Freenet Index come up.

Establishing Your Node

Actually getting your node established in the network can be a little tricky. It should, and usually will, do so automatically, but it often helps to push the process along a little to speed it up. This *is* an optional procedure. There are two main parts to this: finding other nodes and getting known by other nodes. Note that if you're running a transient node the second part isn't important.

The first part is simple, use Freenet. Just take a look at some of the content, such as the above-mentioned Aardvark's Freenet Index (`KSK@Aardvark`) and `gj`'s web page (`KSK@webpages/gj_jump0`).

The second part is a little bit harder. You can insert a random chunk of data (1K) into Freenet with:

```
dd if=/dev/urandom bs=1024 count=1 | freenet_insert -length 1024 CHK@
```

When your node inserts that data, other nodes will find out about your node from the `SourceAddress` on the packets of data.

In any case, remember that it will take a few days before your node becomes known and other nodes start to request data from you. So if nothing much seems to be happening—don't panic!

Day-to-Day Administration

Properly set up, a Freenet node shouldn't need any ongoing administration. However, there are a few automatable tasks that need to be done. First of all, the log files need to be rotated. Secondly, you'll want to restart the Freenet software periodically. Currently the disk (and I believe memory) usage of Freenet tends to balloon unless you restart your node periodically. Both rotating the log files, which requires restarting the node anyway, and restarting the node can be easily done with the script `restart_script` (shown in Listing 1). For your average node, restarting once a week should be fine.

Listing 1. restart script

Recommended JDKs

IBM's JDK is probably the easiest JDK to install. It's precompiled and works, plus it's one of the faster JDKs out there. On the downside it's proprietary and requires registration to download. Both tarball and Red Hat RPMs are available at: www.ibm.com/java/jdk/linux130/index.html.

Kaffe, the OpenSource JDK, is another, more difficult option. You will have to compile in gmp support, something most distributions don't do in their Kaffe packages. A Debian package for Kaffe with gmp support is available from Mr. Bad's site, above. Otherwise, compile from anonymous CVS or a source package to which you apply the patch found in the Freenet README. Make sure you enable gmp support with the `--enable-gmp` option when you run `./configure`.

Resources



Peter Todd, who will be 16 by the time this article is published, has been using Linux for three years (exclusively for one and a half years). He attends Woburn Collegiate where he can be found in Ms. Plachta's grade 12 enriched computer science class and in the music hall. He can be contacted at retep@penguinpowered.com and has a web site at <http://retep.tripod.com/>.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

oftpd: a Secure, Modern FTP Daemon

Don Marti

Issue #84, April 2001

Running an efficient FTP server requires simplicity, security and high performance—take a look at this daemon.

FTP daemons may not get much attention, but what attention they do get is in the form of security advisory after security advisory. There's not much hack value for most people in supporting something as old-school as FTP, so the state of the art has languished—until now. This article will introduce you to Shane Kerr's **oftpd**, an incredibly simple FTP daemon that, we think, will start to replace the old, full-featured daemons currently shipping on (and creating security issues for) Linux distributions everywhere.

Rick Moen, a Linux security guru who is also famous for his ability to install Linux on many obsolete or unusual systems, runs an FTP server mostly as an archive of Linux distributions to install over the network. Boot from floppy, select "Install from FTP" and Bob's your uncle. No need to figure out how to **insmod** the driver for somebody's ten-year-old 1x CD-ROM drive. (Please, don't everyone go knocking on Rick's door with your VIC-20s and PDP-8s; he gets plenty of challenges from regular installfests and user-group meetings.)

FTP remains useful for three reasons, Rick says. First, FTP provides automatic, informative directory listings, including file modification times, to facilitate mirroring. Second, there's no "index.html" to override a daemon-generated directory listing in FTP, so it's easy to mirror entire directory trees or grab them with **snarf**.

Finally, Rick finds that you can get small FTP clients for legacy Oses or special situations where an HTTP client won't fit. For example, Rick used a MacOS FTP client to copy Debian floppies to an old Macintosh with no CD-ROM and insufficient memory for a web browser. People who install a lot of Red Hat or Red Hat-based systems using Kickstart floppies soon find that FTP installs are fast, convenient and let you keep a single software archive up-to-date on the

FTP server. You can do the same thing with an NFS (Network File System) install, but FTP is faster.

The qualities people need from an FTP dæmon have changed since the old dæmons were written. First, the days of logging in to an FTP site with a username and password (which get sent in plaintext over the Net) are gone, gone, gone. People are using **ssh**, **scp** and other encrypted tools to protect their passwords on the Net when they transfer nonpublic files, so the FTP dæmon no longer needs to authenticate users. Second, performance matters. People aren't just using FTP to get an occasional piece of software or pass along work to colleagues. FTP servers should be prepared to get slammed by many clients all installing an entire Linux distribution at once.

So, the two goals for running a modern FTP server are: for security, simplify both your Policy and FTP dæmon to “anonymous FTP only”, and look for a dæmon that offers good performance. That's what brings us to **oftpd**.

For security, **oftpd** runs as a non-root user except for essential setup tasks. It runs “chrooted” to the public FTP directory. These features buy you a lot, security-wise, and an old-fashioned FTP dæmon that allows users to log in with a username and password that can't match it. For simplicity and performance, **oftpd** itself generates directory listings instead of running **ls**.

Moving from Old-School FTP to **oftpd**

The first step in setting up **oftpd** is to get rid of the old **ftp** entry in **inetd.conf**. The FTP dæmon that probably came with your Linux distribution gets run by **inetd**, while **oftpd** runs standalone for speed. Just comment out the **ftp** line with a **#** at the beginning, so it looks something like this:

```
# ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
```

Now **killall -HUP inetd** to restart **inetd**, so that it knows not to respond to connections on the FTP port. If you **ftp localhost** you should get a **Connection refused**.

Now, we're ready to move on to **oftpd** itself. First, the easy part. To compile and install, just unpack the tar file, **cd** into the resulting directory and use the One True GNU Way (**./configure; make; make install**). To keep everything in its appropriate, canonical place, supply the **bindir** argument to the **configure** script, so that **make install** will put the **oftpd** executable in **/usr/local/sbin**, which is the appropriate directory for locally built dæmons:

```
sh ./configure --bindir=/usr/local/sbin && make
```

Then **su** root and do:

```
make install
```

The installation is so generic that the INSTALL document in the version of `oftpd` I used begins simply, “These are generic installation instructions”. That’s a good thing. By the time you read this, there will probably be RPM or deb packages for your favorite distribution, so check there first for an even easier install.

The above compile and install step, as well as the following three steps, will be handled for you if you install `oftpd` from an RPM or deb package, but you’ll have to complete them if you’re installing `oftpd` from source.

Add the `oftpd` User

One of the most important rules of Linux security is “Don’t run anything as root that you don’t have to”. `oftpd`, like Apache and other daemons, follows this rule by dropping root privilege right after it starts listening on its standard, root-only port. If you’re looking at the `oftpd` source, this is found in `src/oftpd.c`, where “create our main listener” is followed immediately by “set user to be as inoffensive as possible”. So, you’ll be starting `oftpd` as root, but you’ll need to add a user that it can run as afterward. Run **`adduser`** or **`useradd`** (whichever one your distribution provides) to create a new user called “`oftpd`”.

Test `oftpd` by Starting and Stopping It Manually

As root, start `oftpd` like this:

```
/usr/local/sbin/oftpd oftgd /home/httpd/html
```

The two arguments are: first, the user to run as and, second, the FTP directory to use. You can point both Apache and `oftpd` at the same directory so that all of your content is available either by HTTP or FTP. Substitute your own web server’s “DocumentRoot” directory as the second argument. You can always make a separate FTP directory if you want, but this way lets people use HTTP to get files from your server if they’re behind a misconfigured firewall that doesn’t let FTP connections through.

If `oftpd` starts without errors, try visiting localhost with your favorite FTP client or **`ftp://localhost/`** with your web browser. You should be able to get a directory listing of your chosen FTP directory. If you do, you’re almost done. Have a refreshing beverage. Two bad things can happen with the above command, but they’re easy to deal with. If you get an “invalid user name” error, you didn’t create the `oftpd` user correctly. If you specify a nonexistent directory, you won’t immediately see an error message, but you won’t get a directory listing, and `oftpd` will log an error using `syslog`. On my Debian box, `syslog` puts `oftpd`’s log messages in `daemon.log`. You can change that by editing `/etc/syslogd.conf`, but that’s another article.

As root, kill oftpd for now with:

```
killall oftpd
```

Make an oftpd init Script

Next, you'll need to make an init script to start and stop oftpd. As root, cd into your init.d directory (/etc/rc.d/init.d on Red Hat, /etc/init.d on Debian) and pick out an init script to copy and edit (I like to copy new init scripts from the one for **sshd**, because it's very simple). Two small features to watch for: oftpd (version 0.2.0) does not currently write a pidfile in /var/run, and the PID of the running process is not the same as the PID of the process you started because oftpd forks when it starts up. So in the "stop" section of your init script, you'll need to take the appropriate steps to kill oftpd by name instead of by PID. On Debian, use the --exec option of **start-stop-daemon** (see Listing 1). However, the **killproc** function on Red Hat will automatically kill by name if it can't find a pidfile.

Listing 1. oftpd init Script for Debian

Welcome Back, Installers of Packages

If you installed a package instead of source, everything up to this point will have been done for you. Check that the init script points oftpd to the appropriate directory, but that's about it. Now, whether you installed from source or from a package, just **cd** into your init.d directory and do:

```
./oftpd start
```

Then check ftp://localhost/ to make sure oftpd is up; use your distribution's runlevel management tool to set oftpd to start automatically when you enter your default runlevel; and **ftp** to your server from the outside to make sure some firewall weenie hasn't filtered FTP out of your network. Now you have a working, simple FTP server suitable for software archives of all kinds. You can mirror your favorite Linux distribution and invite everyone you know to bring their boxes over for a quick network install.

Resources



Don Marti is the technical editor for *Linux Journal*. He can be reached at info@linuxjournal.com.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Linux on Carrier Grade Web Servers

Ibrahim Haddad

Makan Pourzandi

Issue #84, April 2001

Ibrahim and Makan describe and test the Linux Virtual Server.

ARIES (Advanced Research on Internet E-Servers) is a project that started at Ericsson Research Canada in January 2000. It aimed at finding and prototyping the necessary technology to prove the feasibility of a clustered internet server that demonstrates telecom-grade characteristics using Linux and open-source software as the base technology. These characteristics feature guaranteed continuous availability, guaranteed response time, high scalability and high performance.

Traffic distribution was one of the main research topics because we needed a software-based solution that was very reliable, high performance and yet very scalable to distribute web traffic among multiple CPUs within the cluster. The first activity we conducted was to survey the Open Source community and see what solutions were already implemented, try them out on our experimental Linux cluster and see which solution best met our requirements (or part of them).

This article covers our experience with the Linux Virtual Server (LVS), a software package that provides traffic distribution on top of Linux. We will explain the architecture, operation and algorithms of LVS and describe our test system and benchmarking environment as well as the experiments we conducted to test LVS's robustness and scalability.

The traffic on the Internet is growing at over 100% every six months. Thus, the workload on the servers is increasing rapidly, and these servers are easily overloaded.

To overcome this problem there are several solutions. The most interesting one is the multiserver solution that consists of building a scalable server on a cluster of servers. When the load increases, more servers can be added into the cluster to meet the increasing requests. The multiserver architecture provides good flexibility (adding and removing nodes) and availability since there is no downtime associated with it. A second advantage is that the real servers do not need to be homogeneous, which allows for the recycling of some old workstations. A third advantage is transparency: the cluster is presented to the users as a single IP that maps requests to multiple servers at the back end.

The Linux Virtual Server follows the multiserver model and provides a software-based solution for traffic distribution as opposed to hardware-based solutions.

Our research in the ARIES Project was directed toward a multiserver architecture that is able to achieve linear scalability reflected through a continuous growth to meet increasing demands, continuous service availability achieved through building redundancy at all levels of the architecture and ease and completeness of management without affecting the uptime of the system.

We were naturally interested in LVS as an HTTP traffic distribution solution because the architecture of the cluster will be transparent to end users, and thus the whole system would appear with a single IP address. Furthermore, LVS claims to be highly available by detecting node or daemon failures and reconfiguring the system appropriately so that the workload can be taken over by the remaining nodes in the cluster. This is a very important feature for systems geared toward high availability.

The LVS Project is an open-source project to cluster many servers together into a highly available and high-performance virtual server that provides good scalability, reliability and serviceability. The LVS director provides IP-level load balancing to make parallel services of the cluster appear as a virtual service on a single IP address.

The LVS Project is currently cooperating with the High Availability Linux Project that aims at providing a high-availability (clustering) solution for Linux, which promotes reliability, availability and serviceability (RAS) through a community-development effort.

LVS Architecture

In an LVS setup, the real servers may be interconnected by a high-speed LAN or by geographically dispersed WAN. On the other hand, the front end of the real servers, called director, is a load balancer that distributes requests to the different servers. All requests are sent to the front-end director with the virtual IP address, and the cluster appears as a virtual service on a single IP address.

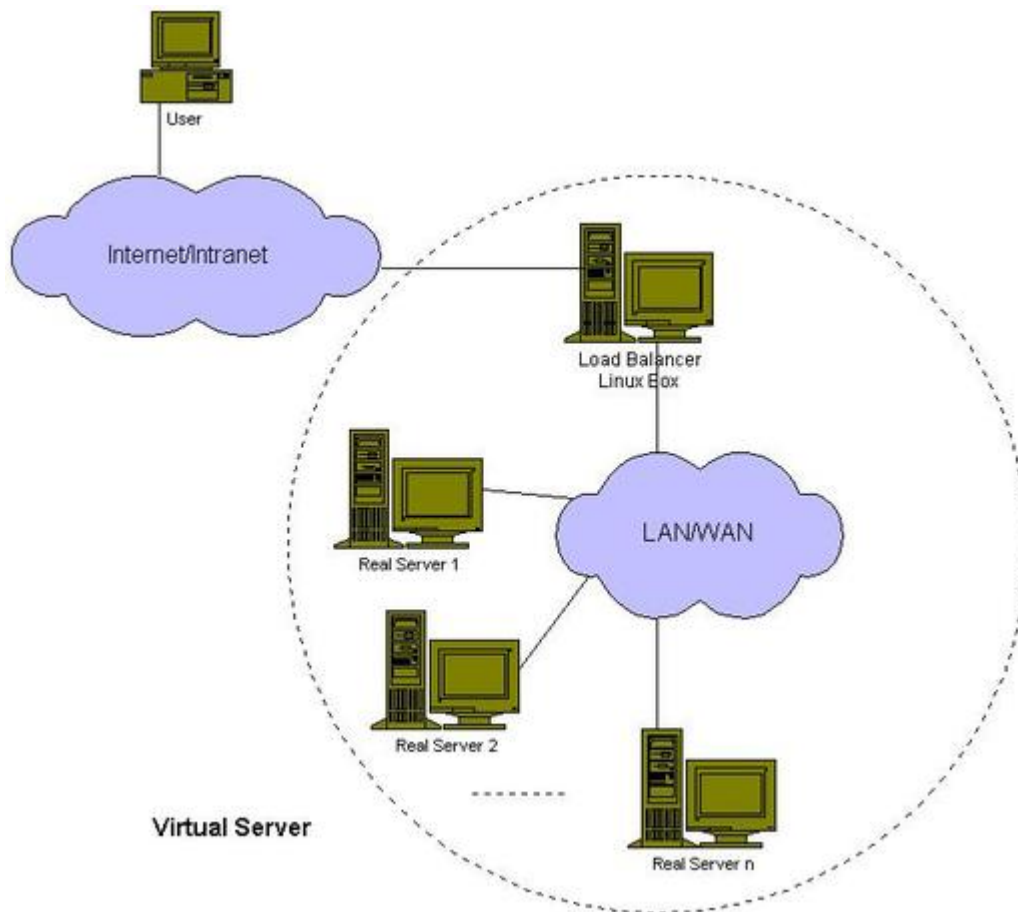


Figure 1. LVS Architecture

This architecture is flexible because it allows transparently adding or removing real server nodes, and it is geared toward high availability by automatically detecting nodes or daemons failures and reconfiguring the system appropriately. For added availability, we can setup a second director as a hot swap for the primary director, thus eliminating a possible single point of failure.

LVS is implemented in three IP load-balancing techniques. One is virtual server via network address translation (NAT), the second is virtual server via IP tunneling and the third is virtual server via direct routing. When we conducted this activity, the NAT implementation was very stable compared to the others. Therefore, we decided to setup LVS using NAT.

Virtual Server via NAT

The LVS-NAT implementation was necessary due to the large usage of private addresses. For many reasons, including the shortage of IP addresses in IPv4 for security reasons, more networks are using private IP addresses that cannot be used outside the network.

Network address translation relies on the fact that the headers for internet protocols can be adjusted appropriately so that clients believe they are contacting one IP address, but servers at different IP addresses believe they are

contacted directly by the clients. This feature can be used to build a virtual server, i.e., parallel services at the different IP addresses can appear as a virtual service on a single IP address via NAT.

The architecture of a virtual server via NAT is illustrated in Figure 2. The load balancer and real servers are interconnected by a switch or a hub. The real servers usually run the same service, and they have the same set of contents. The contents are replicated on each server's local disk, shared on a network filesystem or served by a distributed filesystem. The load balancer dispatches requests to different real servers via NAT.

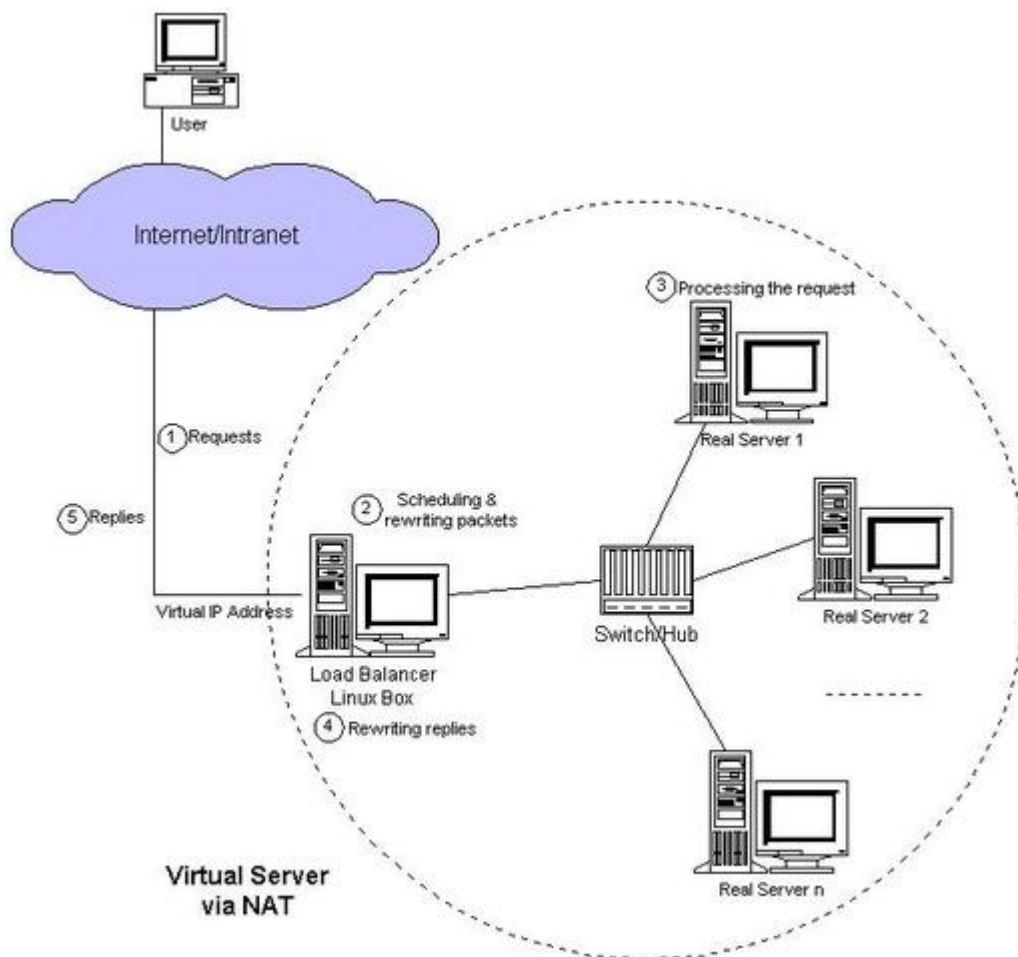


Figure 2. Virtual Server via NAT

The Process of Address Translation

The process of address translation is divided into several steps:

1. The user accesses the service provided by the server cluster.
2. The request packet arrives at the load balancer through the external IP address of the load balancer.
3. The load balancer examines the packet's destination address and port number. If they match a virtual server service (according to the virtual

server rule table), then a real server is chosen from the cluster by a scheduling algorithm, and the connection is added into the hash table, which records all established connections.

4. The destination address and the port of the packet are rewritten to those of the chosen server, and the packet is forwarded to the real server.
5. When the incoming packet belongs to this connection, and the established connection can be found in the hash table, the packet will be rewritten and forwarded to the chosen server.
6. When the reply packets come back from the real server to the load balancer, the load balancer rewrites the source address and port of the packets to those of the virtual service.
7. When the connection terminates or timeouts, the connection record will be removed from the hash table.

Scheduling Algorithms

LVS supports four scheduling algorithms: round-robin, weighted round-robin, least-connection scheduling and weighted-least-connection scheduling. The round-robin algorithm directs the network connections to the different server in a round-robin manner. It treats all real servers as equals regardless of number of connections or response time. The weighted round-robin scheduling is able to treat the real servers of different processing capacities. Each server can be assigned a weight and integer value, indicating its processing capacity, and the director schedules requests depending on the server's weight in a round-robin fashion.

The least-connection scheduling algorithm directs network connections to the server with the least number of established connections. This is a dynamic scheduling algorithm because it needs to count live connections for each server dynamically. At a virtual server where there is a collection of servers with similar performance, the least-connection scheduling is good for smoothing distribution when the load of requests varies a lot because all long requests won't have a chance to be directed to a server.

The weighted least-connection scheduling is a superset of the least-connection scheduling in which you can assign a performance weight to each real server. The servers with a higher weight value will receive a larger percentage of live connections at any one time. The virtual server administrator can assign a weight to each real server, and network connections are scheduled to each server in which the percentage of the current number of live connections is a ratio to its weight. The default weight is one.

Testing Environment

For the purpose of testing and evaluating LVS's performance, we setup a powerful testing environment at the Ericsson Research Lab in Montréal that consisted of eight CompactPCI diskless Pentium III CPU cards running at 500MHz with 512MB of RAM. The CPUs have two on-board Ethernet ports, and each CPU is paired with a four-port ZNYX Ethernet card. Our setup also included a CPU with the same configuration as the others except that it has three SCSI disks: 18GB each configured with a RAID-1 and RAID-5 setup. This CPU acts as the NFS, DHCP, Bpbatch and TFTP server for the other CPUs.

Operation of the CPUs

The CPUs use the Linux kernel 2.2.14-5.0 that comes with the Red Hat 6.2 distribution. When we start the CPUs, they boot from LAN and broadcast a DHCP request to all addresses on the network. The DHCP server (the CPU with disk) will reply with a DHCP offer and will send the CPUs the information they need to configure network settings such as IP-addresses (one for each interface, eth0 and eth1), gateway, netmask, domain name, the IP address of the boot server and the name of the boot file. Using **bpbatch**, a freeware preboot software, the diskless CPUs will then download and boot the specified boot file in the bpbatch configuration file. The boot file is a kernel image located under the /tftpboot directory on the bpbatch server. Finally, the CPUs will download a RAM disk and start the Apache web server.

The Apache web server is part of the RAM disk that is downloaded by the CPUs. Because we have a homogeneous environment, all the CPUs share the same configuration files and serve the same content, which is available via NFS.

Benchmarking Environment

To conduct benchmarking activities, we setup a hardware and a software benchmarking environment. At the hardware level, we used 18 Intel Celeron 500MHz 1U rackmount units with 512MB of RAM each. These machines were used to generate web traffic using **WebBench**, a freeware tool available from znet.com.

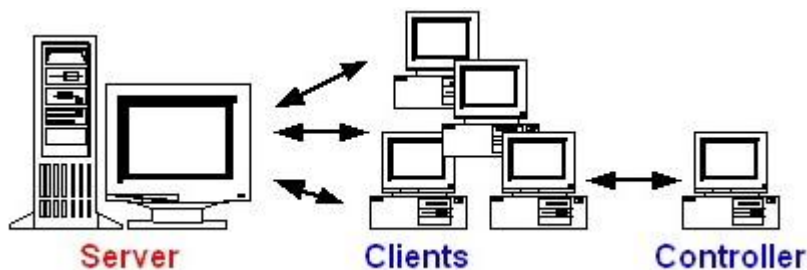


Figure 3. WebBench Architecture

WebBench is a software tool for measuring the performance of web servers. It consists of one controller and many clients (Figure 3). The controller provides means to setup, start, stop and monitor the WebBench tests. It is also responsible for gathering and analyzing the data reported from the clients (Figure 4). On the other hand, the clients execute the WebBench tests and send requests to the server.

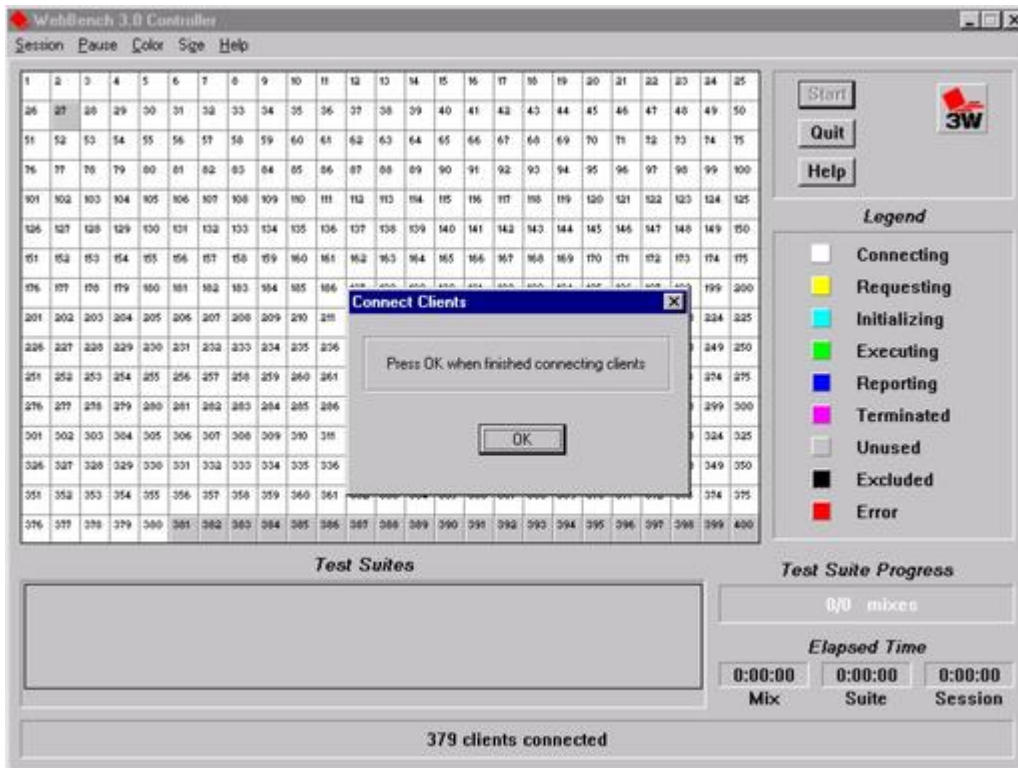


Figure 4. WebBench Control Window with 379 Connected Clients Ready to Generate Traffic

Building the LVS-NAT Setup

To build the setup, we decided to use the CPU with disk as the Linux Virtual Server and the eight diskless CPUs, running the Apache server, as traffic processors.

As mentioned previously, we were using the Linux kernel supplied with Red Hat 6.2. This kernel comes with LVS support, and thus there was no need to do any work at the kernel level. We only needed to perform the configuration. For those who like GUI configuration tools, they will find the LVS GUI-configuring tool a definite advantage. It is a complete tool to setup and manage an LVS environment, and it is easy to use. However, we performed our configuration manually—a matter of personal preference.

If you are not using a kernel provided with Red Hat (6.1 or later), then you need to build a new kernel and setup IP masquerading and IP chain rules manually. The kernel you build must have support for the following options: network

firewalls, IP forwarding/gatewaying, IP firewaling, IP masquerading, IP: ipportfw masq and the Virtual Server support.

At the same time, you need to choose a scheduling algorithm from among the following: weighted round-robin, least-connection or the weighted-least connection.

Once this is done, you compile the kernel, update your system and reboot. After rebooting with the new kernel, you need to setup the IP configuration for the NAT director, which includes setting an alias IP address to be used for access from outside the cluster and setting an alias for the NAT router for access from inside the cluster and enabling `ip_forward` and `ip_always_defrag` on NAT director.

When we completed these steps, we configured LVS by editing the `lvs.conf` file and started the `lvs` binary that came with Red Hat. It performed the necessary `ipvsadm` commands for the LVS server and the real servers.

Please note that if you are not using the Red Hat distribution, then you need to start the `ipvsadm` commands manually. After that, you need to start `ipchains` with the `forward` and `MASQ` options.

Setup Tip

If you plan to deploy LVS in a LAN environment, you have to be careful when isolating the real servers from the rest of the LAN and from the outside world. There must be no direct route from any real server to the web clients. This is very important because if the HTTP answer packets go to the client through a way other than the NAT director, they will be discarded by the client. One easy way to achieve this is to build your own private subnet and use the NAT director as your gateway to the rest of the LAN.

Benchmarking Scenarios

The goal of the benchmarking tests was to test the scalability of the LVS-NAT implementation. For this purpose, we carried two kinds of tests: the first one was a direct approach that consisted of sending traffic directly to the CPUs; the second approach was to direct the traffic to the NAT director that is the front-end server for the CPUs.

For the tests conducted without LVS, we sent HTTP requests directly to the real servers. WebBench supports this configuration by generating web traffic and sending them to multiple servers (Figure 5). As for the tests with LVS (Figure 6), we configured WebBench to send the HTTP requests to the LVS server (the NAT director), which in turn directed the traffic to the real servers.

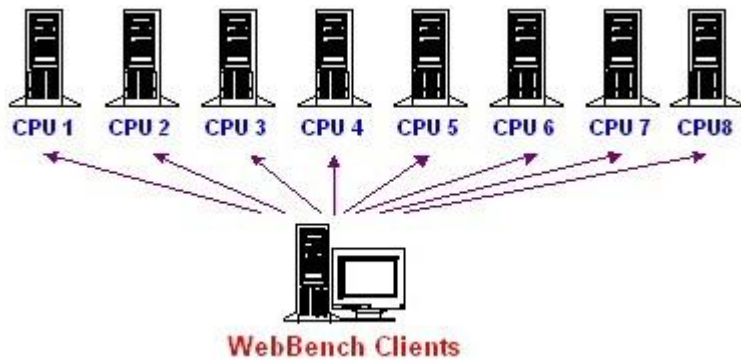


Figure 5. The Benchmarking Setup without LVS

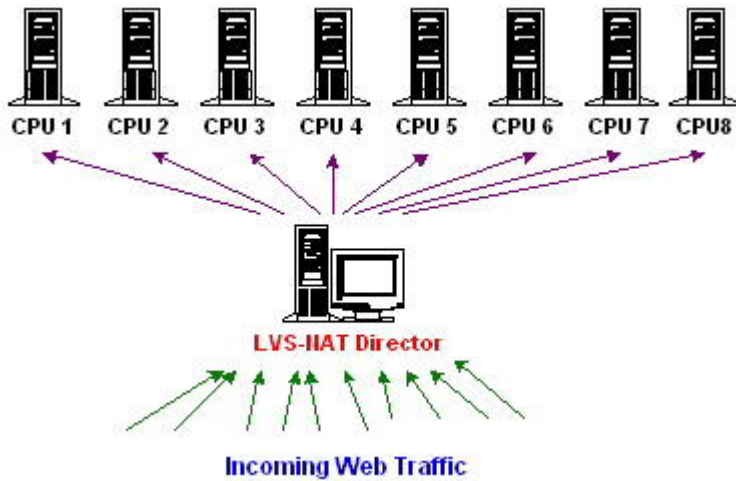


Figure 6. The Benchmarking Setup with LVS-NAT

Benchmarking Results

Figure 7 shows the number of requests per second our LVS setup was able to achieve versus a direct setup without LVS. It clearly shows that the LVS-NAT implementation suffers from a bottleneck at the director level once it reaches 2,000 requests per second.

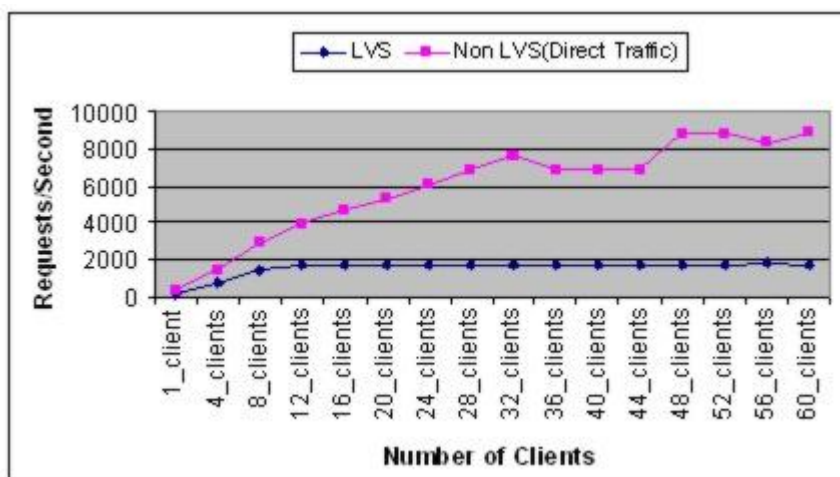


Figure 7. LVS vs. Non-LVS Results

We decided to conduct a third test using only one machine to generate traffic with WebBench. We measured a latency of 0.3 milliseconds for answering HTTP requests. LVS handled the load successfully answering more than 178 requests per second.

After analyzing the results, we concluded that the bottleneck problem is due to the number of simultaneous TCP connections per second that the LVS director can handle. The results show that the maximum number of connections handled simultaneously by LVS is not more than 1,790 valid TCP connections per second. Without using LVS, by sending requests directly to servers, we have been able to achieve more than 7,116 valid TCP connections per second. We plan to investigate this issue in more detail in the coming weeks.

Evaluation of LVS via NAT

The NAT implementation of LVS has several advantages. First, the real servers can run any operating system that supports the TCP/IP protocol and they can use private internet addresses. As a result, the whole setup would only require one IP address for the load balancer.

However, the drawback of using the NAT implementation is the scalability of the virtual server. As we have seen in the benchmarking tests, the load balancer presents a bottleneck for the whole system.

LVS via NAT can meet the performance request of many small to mid-size servers. When the load balancer becomes a bottleneck, then you need to consider the other two methods offered by LVS: IP tunneling or direct routing.

Conclusion

We tested LVS in an industrial environment with one LVS Server and eight traffic CPUs. We found some restrictions when using LVS under heavy load. However, we also found LVS to be easy to install and manage and very useful.

We believe that LVS is a potential solution for small to mid-size web farms that need a software-based solution for traffic distribution. However, for the kind of servers we are building, the requirements necessitate a higher number of transactions per second than the NAT implementation of LVS can handle.

LVS's future is promising with the determination to add more load-balancing algorithms and geographic-based scheduling for the virtual server via IP tunneling. Another promising future feature is the integration of the heartbeat code and the CODA distributed fault-tolerant filesystem into the virtual server. LVS's developers are also planning to explore higher degrees of fault-tolerance and how to implement the virtual server in IPv6.

Compared to other packages, LVS provides many unique features such as the support for multiple-scheduling algorithms and for various methods of requests forwarding (NAT, direct routing, tunneling). Our next step regarding LVS is to try out the other two implementations (direct routing and IP tunneling) and compare the performance with the NAT implementation on the same setup.

Acknowledgments

- The Systems Research Department at Ericsson Research Canada for approving the publication of this article.
- Evangeline Paquin, Ericsson Research Canada, for her contributions to the overall LVS-related activities.
- Marc Chatel, Ericsson Research Canada, for his considerable help in the ECUR Lab.
- Wensong Zhang, the LVS Project, for the permission to use Figures 1 and 2 from the LVS web site.

Resources



Ibrahim Haddad (ibrahim.haddad@lmc.ericsson.se) works for Ericsson Research Canada in the Systems Research Department researching carrier class server nodes in real-time all-IP networks. He is currently a DrSc candidate in the Computer Science Department at Concordia University in Montréal, Canada.



Makan Pourzandi (makan.pourzandi@lmc.ericsson.se) works for Ericsson Research Canada in the Systems Research Department. His research domains are security, cluster computing and component-based methods for distributed programming. He received his Doctorate in Parallel Computing in 1995 from the University of Lyon, France.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Managing Initscripts with Red Hat's **chkconfig**

Jimmy Ball

Issue #84, April 2001

A simple but quite useful addition to your administration command vocabulary.

I love discovering new UNIX commands, especially those with a system administration flavor. When I learned Red Hat distributed the **chkconfig** utility, it brought back fond memories of **chkconfig** under IRIX, a UNIX variant from Silicon Graphics, Inc. IRIX's **chkconfig** was designed to enable/disable services for automatic launch during system initialization without editing, renaming or moving initscripts in **/etc**.

Similarly, Red Hat designed **chkconfig** to help manage services launched during system initialization. But, after perusing the man page and doing some tests, I soon found that Red Hat extended **chkconfig** with finer control of system startup/shutdown tasks by managing the symbolic links to initscripts. It's a real time-saver!

Startup Basics

When your Linux box boots, the first process that shows up is **init**. If you haven't seen **init** before, take a moment to type **ps -ef | grep init** to see the PID of **init**. In short, the **init** performs tasks that are outlined in **/etc/inittab**.

Some tasks outlined in **/etc/inittab** will be launched soon after **init**, while others are simply set up. For example, the default Red Hat **/etc/inittab** sets up a trap for the key sequence Ctrl-Alt-Delete. When these keys are simultaneously pressed at a console prompt (not **xdm**), the shutdown command is performed. At boot time, **init** sets up this feature based on configuration options in **/etc/inittab**, but execution is postponed until the key sequence occurs.

The format of **inittab** allows for comment lines beginning with a **"#"** symbol while normal entries are **":"** delimited. They follow the pattern **id:runlevel:action:process** where **id** represents a user-defined and unique

identifier, runlevel can be a combination of the numbers 0-6 or just left blank, action comes from a keyword that describes how init should treat the process, and process is the command to execute.

Descriptions of various keywords for the action field can typically be found in the man pages for inittab. Common keywords across most, if not all, UNIX platforms include:

- `initdefault`—defines the runlevel to enter once the system has booted.
- `wait`—a process that will be executed once (when the runlevel is entered). The init process will wait for this process to terminate.
- `boot`—defines a process that is executed at boot time.
- `bootwait`—similar to `boot` but init waits for the process to terminate before moving on.
- `sysinit`—defines a process that is executed at boot time before any `boot` or `bootwait` inittab entries.

The runlevel field designates system state. For example, a runlevel of 0 corresponds to a halted system while a runlevel of 6 corresponds to a system reboot. Unfortunately, all Linux distributions do not follow the same definition for runlevels. Under Red Hat, the following defaults are supported:

```
0. System halt
1. Single-user mode
2. Multiuser, without NFS
3. Complete multiuser mode
4. User defined
5. X11 (XDM login)
6. Reboot
```

For each runlevel, there is a corresponding directory in `/etc/rc.d`. For a runlevel of 5, the directory `/etc/rc.d/rc5.d` exists and contains files related to tasks that need to be performed when booting into that runlevel. Under Red Hat, these files are typically symbolic links to shell scripts found in `/etc/rc.d/init.d`.

Let's put this all together with a simple example. Below are two sample lines from our inittab file:

```
id:3:initdefault:
l3:3:wait:/etc/rc.d/rc 3
```

Here is a typical scenario of what happens under Red Hat. Once init is started, it reads `/etc/inittab` (see above). From the first line, we know that init is going to end up at a runlevel of 3 after the system boots. Once we reach that runlevel, the second line tells init to run the script `/etc/rc.d/rc` three and waits for it to terminate before proceeding.

The script `rc` in `/etc/rc.d` receives `3` as an argument. This `3` corresponds to a runlevel of `3`. As a result, the `rc` script executes all the scripts in the `/etc/rc.d/rc3.d` directory. It first executes all the scripts that begin with the letter `K` (meaning “kill” the process or service) with an argument of “stop”. Next, it runs all the scripts that begin with the letter `S` with an argument of “start” to start the process or service. As one final note, the order of `K` and `S` script execution is based on sort order; the script named `S90mysql` would execute before the script named `S95httpd`.

It turns out the scripts in `/etc/rc.d/rc3.d` are actually symbolic links to scripts residing in `/etc/rc.d/init.d`. While the UNIX administrator can place scripts in `rc3.d`, the common practice under Red Hat is to first place all scripts in `init.d`, then create logical links to the `rc*.d` directories. It doesn't take long to figure out the creation and maintenance of these scripts and symbolic links could be quite the chore. That's precisely where `chkconfig` steps in! The Red Hat `chkconfig` utility is specifically designed to manage the symbolic links in `/etc/rc.d/rc[0-6].d`.

Viewing `chkconfig` Entries

The `chkconfig` binary resides in `/sbin` with default permissions that allow any user to execute it, although the user without root privileges can only view the current `chkconfig` configuration. So type

```
[root]# chkconfig --list | grep on
```

A partial listing of the output would look similar to the following:

```
amd 0:off 1:off 2:off 3:off 4:on 5:on 6:off
apmd 0:off 1:off 2:on 3:off 4:on 5:off 6:off
arpwatch 0:off 1:off 2:off 3:off 4:off 5:off 6:off
atd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
autofs 0:off 1:off 2:off 3:off 4:off 5:off 6:off
named 0:off 1:off 2:off 3:off 4:off 5:off 6:off
bootparamd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
keytable 0:off 1:off 2:on 3:on 4:on 5:on 6:off
crond 0:off 1:off 2:on 3:on 4:on 5:on 6:off
syslog 0:off 1:off 2:on 3:on 4:on 5:on 6:off
netfs 0:off 1:off 2:off 3:on 4:on 5:on 6:off
network 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

On each line of output, the first field represents the name of an initscript in `/etc/rc.d/init.d`. The remaining fields correspond to the runlevels `0-6` along with the status of the script when entering that runlevel. For example, **crond** would be launched when entering the runlevels `2, 3, 4` and `5` and stopped when entering the runlevels `0, 1` and `6`. We can confirm that these settings are true with the **find** command to search for all files in `/etc/rc.d` that end with `crond`:

```
[root]# find /etc/rc.d -name '*crond' -print
/etc/rc.d/init.d/crond
/etc/rc.d/rc0.d/K60crond
/etc/rc.d/rc1.d/K60crond
/etc/rc.d/rc2.d/S40crond
```

```
/etc/rc.d/rc3.d/S40crond
/etc/rc.d/rc4.d/S40crond
/etc/rc.d/rc5.d/S40crond
/etc/rc.d/rc6.d/K60crond
```

Notice for each “off” section reported by `chkconfig` (0, 1, 6), a kill script is in place and for each “on” section reported (2, 3, 4, 5), a start script exists. Next, execute a different `find` command to determine the type of each file found:

```
[root]# find /etc/rc.d -name '*crond' -exec file {} \;
/etc/rc.d/init.d/crond: Bourne shell script text
/etc/rc.d/rc0.d/K60crond: symbolic link to
../init.d/crond
/etc/rc.d/rc1.d/K60crond: symbolic link to
../init.d/crond
/etc/rc.d/rc2.d/S40crond: symbolic link to
../init.d/crond
/etc/rc.d/rc3.d/S40crond: symbolic link to
../init.d/crond
/etc/rc.d/rc4.d/S40crond: symbolic link to
../init.d/crond
/etc/rc.d/rc5.d/S40crond: symbolic link to
../init.d/crond
/etc/rc.d/rc6.d/K60crond: symbolic link to
../init.d/crond
```

This reveals that `crond` found inside `init.d` is a shell script and all remaining files found are symbolic links to the `crond` script.

Modifying `chkconfig` Entries

Modifying a `chkconfig` entry is almost as easy as listing the existing configuration. The form is:

```
chkconfig [--level <levels>] <name> <on|off|reset>
```

For example, if we decide to disable `crond` for runlevel 2, the **`chkconfig --level 2 crond off`** command (executed by `root`) would turn off `crond` for the runlevel of 2. Running **`chkconfig --list`** will confirm that `crond`'s configuration has been modified. Further, the `find` command below reveals that a kill script has replaced the start script in the `rc2.d` directory:

```
[root]# find /etc/rc.d -name '*crond' -print
/etc/rc.d/init.d/crond
/etc/rc.d/rc0.d/K60crond
/etc/rc.d/rc1.d/K60crond
/etc/rc.d/rc2.d/K60crond
/etc/rc.d/rc3.d/S40crond
/etc/rc.d/rc4.d/S40crond
/etc/rc.d/rc5.d/S40crond
/etc/rc.d/rc6.d/K60crond
```

Keep in mind that `chkconfig` does not automatically disable or enable a service immediately. It simply changes the symbolic links. The superuser could disable the `crond` service immediately with the command `/etc/rc.d/init.d/crond stop`. Finally, you can enable/disable a command for multiple runlevels with one `chkconfig` command. For example entering

```
chkconfig --levels 2345 crond on
```

would set up crond to be started for runlevels 2, 3, 4 and 5.

Removing an Entry

At times, removing a service altogether may be in order. Take sendmail, for example. On client machines where incoming mail for local accounts is not required, running sendmail as a daemon may not be necessary. In this case, I find disabling sendmail desirable since it reduces potential security risks. To remove sendmail from chkconfig, type

```
chkconfig --del sendmail
```

Below, our find command shows no symbolic links in place, while the initscript for sendmail remains:

```
[root]# find /etc/rc.d -name '*sendmail' -print  
/etc/rc.d/init.d/sendmail
```

This is perfect in my opinion. The script is left in case sendmail needs to be re-established as a service, but all symbolic links are gone. While we could have disabled sendmail for all runlevels, this would have placed kill scripts in each of the rc*.d subdirectories, an unnecessary task since sendmail was never launched during the initialization phase. However, I have seen situations when the system administrator would manually start a service on certain occasions. Having the kill scripts in place for that service ensures a cleanly killed service. So, you make the call.

Adding a chkconfig Entry

So far, so good. We've seen how to view, modify and delete services using chkconfig. It's time to add a new service. Take the script named **oracle** (see Listing 1).

Listing 1. Oracle Script

Using this script, Oracle 8 can be started with the "start" argument and terminated with the "stop" argument. This meets the minimum requirements of an initscript that can be used in conjunction with the launch script /etc/rc.d/rc.

Place the script in /etc/rc.d/init.d and run (as root)

```
chmod +x /etc/rc.d/init.d/oracle
```

to make the script executable. If you are concerned about normal users seeing the script, you could try more restrictive file permissions, as long as the script is executable by root as a standalone script.

Notice the two comments lines in the script:

```
#chkconfig: 2345 80 05
#description: Oracle 8 Server
```

These lines are needed by chkconfig to determine how to establish the initial runlevels to add the service as well as set the priority for the start-and-stop script execution order. These lines denote the script will start Oracle 8 server for the runlevels 2, 3, 4 and 5. In addition, the start priority will be set to 80 while the stop priority will be 05.

Now that the script is in place with the appropriate execute permissions and the required chkconfig comments are in place, we can add the initscript to the chkconfig configuration by typing, as root, **chkconfig --add oracle**.

Using chkconfig's query feature, we can verify our addition:

```
[root]# chkconfig --list | grep oracle
oracle          0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

Also, we can type our standard find command to see how chkconfig set up the symbolic links:

```
[root]# find /etc/rc.d -name '*oracle' -print
/etc/rc.d/init.d/oracle
/etc/rc.d/rc0.d/K05oracle
/etc/rc.d/rc1.d/K05oracle
/etc/rc.d/rc2.d/S80oracle
/etc/rc.d/rc3.d/S80oracle
/etc/rc.d/rc4.d/S80oracle
/etc/rc.d/rc5.d/S80oracle
/etc/rc.d/rc6.d/K05oracle
```

As requested, the names of the kill links contain the priority 05 while the start links contain 80. If we need to adjust the priorities, (e.g., our stop priority needs to be 03), simply modify the chkconfig comment lines in the initscript for oracle and run the **reset** command, as shown below. The resulting symbolic links will be renamed accordingly:

```
[root]# chkconfig oracle reset
[root]# find /etc/rc.d -name '*oracle' -print
/etc/rc.d/init.d/oracle
/etc/rc.d/rc0.d/K03oracle
/etc/rc.d/rc1.d/K03oracle
/etc/rc.d/rc2.d/S80oracle
/etc/rc.d/rc3.d/S80oracle
/etc/rc.d/rc4.d/S80oracle
/etc/rc.d/rc5.d/S80oracle
/etc/rc.d/rc6.d/K03oracle
```

Enhancements in Red Hat 7

As many of you already know, **inetd** was replaced by **xinetd** in Red Hat 7. In addition, chkconfig functionality has been extended to manage some of the functionality of xinetd's Internet services. Sample output is shown below:

```
[root]# chkconfig --list
...
xinetd based services:
  finger: on
  linuxconf-web: off
  rexec: off
  rlogin: off
  rsh: off
  ntalk: off
  talk: off
  telnet: on
  tftp: off
  wu-ftpd: on
```

To disable a xinetd feature, perhaps finger, you could type **[root]# chkconfig finger off**.

Pretty neat, huh? However, there is one “gotcha”. When the configuration is changed, the xinetd is signaled automatically to reload the new configuration with the command `/etc/init.d/xinetd reload`, that is executed by `chkconfig`. This script performs a kill with the SIGUSR2 signal which instructs xinetd to perform a hard reconfiguration.

What does that mean? Well, when I tested it, the active sessions of services offered through xinetd (i.e., Telnet, FTP, etc.) were immediately terminated. That might not be a problem for you, assuming you can plan the best time to disable/enable xinetd services on your system. As an alternative, you can modify the `/etc/init.d/xinetd` script so that the reload option sends a SIGUSR1 signal, which is a soft reconfiguration. This will restart the services without terminating existing connections.

Adding xinetd services for `chkconfig` management is as simple as adding an xinetd service file into the `/etc/xinetd.d` directory. The `chkconfig` utility will automatically pick it up and make it available for management through the `chkconfig` utility. Neat!

Conclusion

Hopefully, you've seen the benefits of Red Hat's `chkconfig` utility for managing initscripts. While its functionality seems simple, the timesaving benefits makes `chkconfig` an administrator's command worth committing to memory.



Jimmy Ball is an instructor with Batky-Howell, Inc. where he teaches UNIX, Perl and Java courses. He can be reached by e-mail at jb@batky-howell.com.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Using Mix-ins with Python

Chuck Esterbrook

Issue #84, April 2001

An explanation of the mix-in programming style as applied in Python.

Mix-in programming is a style of software development where units of functionality are created in a class and then mixed in with other classes. This might sound like simple inheritance at first, but a mix-in differs from a traditional class in one or more of the following ways. Often a mix-in is not the “primary” superclass of any given class, does not care what class it is used with, is used with many classes scattered throughout the class hierarchy and is introduced dynamically at runtime.

There are several reasons to use mix-ins: they extend existing classes in new areas without having to edit, maintain or merge with their source code; they keep project components (such as domain frameworks and interface frameworks) separate; they ease the creation of new classes by providing a grab bag of functionalities that can be combined as needed; and they overcome a limitation of subclassing, whereby a new subclass has no effect if objects of the original class are still being created in other parts of the software.

So while mix-ins are not a distinct technical feature of Python, the benefits of this technique are worth studying.

Python provides an ideal language for mix-in development because it supports multiple inheritance, supports full-dynamic binding and allows dynamic changes to classes. Before we dive into Python, let me admit that mix-ins are old hat. The first time I saw mix-in programming by that name was when reviewing the now-defunct Taligent Project, known for its Pink operating system and CommonPoint application framework. However, since C++ does not support language feature #2, full-dynamic binding, or language feature #3, dynamic changes at runtime, I'm not surprised that the approach didn't bring to fruition all its inventors had hoped for.

I have also seen another instance of mix-in programming under a different name. Objective-C has a nifty language feature called categories that allows you to add and replace methods of existing classes, even without access to their source code.

This is great for repairing existing system classes and extending their capabilities. Also, combined with an ability to load libraries dynamically, categories are quite effective in improving the structure of applications and reducing code.

The grapevine informs me that Symbolics' object-oriented Flavors system is most likely the earliest appearance of bona fide mix-ins. The designers were inspired by Steve's Ice Cream Parlor in Cambridge, Massachusetts where customers started with a basic flavor of ice cream (vanilla, chocolate, etc.) and added any combination of mix-ins (nuts, fudge, chocolate chips, etc.). In the Symbolics system, large, standalone classes were known as flavors while smaller helper classes designed for enhancing other classes were known as mix-ins. A reference can be found on the Web at www.kirkrader.com/examples/cpp/mixin.htm.

Objective-C: I Knew Him Well

Python 2.0

Python Capabilities

Having paid our respects to the dead (Taligent), nearly dead (Objective-C) and legendary (Symbolics), let's start digging into the features that make Python a great language for mix-in programming. For one, Python supports multiple inheritance. That is, in Python, a class can inherit more than one class:

```
class Server(Object, Configurable):  
    pass
```

also, Python supports full-dynamic binding. When passing a message to an object such as:

```
obj.load(filename)
```

Python will determine, entirely at runtime, what method to invoke, based on the name of the message and the class inheritance of obj. This behavior works as expected and is easy to remember. It continues to work even if the class inheritance or method definitions are altered at runtime.

One thing to keep in mind is the order of searching with regard to multiple inheritance. The search order goes from left to right through the base classes, and for any given base class, goes deep into its ancestor classes.

When you create mix-ins, keep in mind the potential for method names to clash. By creating distinct mix-ins with well-named methods you can generally avoid any surprises. Lastly, Python supports dynamic changes to the class hierarchy.

Most Python “things”, whether they are lists, dictionaries, classes or instances, have a set of accessible attributes. Python classes have an attribute named `__bases__`, which is a tuple of their base classes. Consistent with Python design, you can play with it at runtime. In the following session with the Python interactive interpreter seen in Listing 1, we create two classes and then later change the inheritance. Our person in Listing 1 isn't very friendly so let's change it. In fact, let's change all people so that we'll never have this problem again:

```
<<< Person.__bases__ += (Friendly,)
<<< p.hello()
Hello
```

Listing 1. Installing a Mix-in Dynamically

The first statement above changes the base classes of Person. By using `+=` (as opposed to `=`) we avoid accidentally removing existing base classes, especially if a future version of the code makes Person inherit from another class. Also, the funny looking expression, **(Friendly,)**, specifies a tuple that would normally simply be enclosed in parenthesis. However, while Python readily recognizes `<If“Courier”>(x,y)<IF>` as a tuple of two elements, it recognizes `<If“Courier”>(x)<IF>` as a parenthesized expression. Appending the comma forces tuple recognition.

MySQLdb Cursor Mix-ins

The most straightforward way to apply mix-ins is at design time within the construction of a module. One of the more famous third-party modules for Python, MySQLdb, does exactly this.

Python defines a standard programmatic interface for database access named DB API (<http://www.python.org/topics/database/>). Andy Dustman's MySQLdb module implements this interface so that Python programmers can make connections and send queries to a MySQL server. It can be found at <http://dustman.net/andy/python/MySQLdb/>.

MySQLdb provides three major features for the cursor objects it creates. It reports warnings when necessary; it stores result sets on the client side or uses

them on the server side as needed, and it returns results as tuples (e.g., immutable lists) or dictionaries.

Rather than combining all of these into one monolithic class, MySQLdb defines mix-in classes for each of them:

```
class CursorWarningMixin:
class CursorStoreResultMixin:
class CursorUseResultMixin:
class CursorTupleRowsMixin:
class CursorDictRowsMixin(CursorTupleRowsMixin):
```

Remember that mix-ins are classes, so they can take advantage of inheritance, as we see with `CursorDictRowsMixin`, which inherits `CursorTupleRowsMixin`.

None of the mix-ins above can stand on their own: a `BaseCursor` class provides the required core functionality for any type of cursor. Using these mix-ins in combination with `BaseCursor`, MySQLdb offers every combination of warnings, storage and result types (eight in all). When creating a database connection, you can pass the cursor class you desire:

```
conn = MySQLdb.connection (cursorclass=MySQLdb.DictCursor)
```

Mix-ins don't only help in the creation of MySQLdb itself. They also make it more extensible by allowing you to pick and choose features for your own custom cursor classes.

Note that these class names are suffixed with `Mixin` to emphasize their nature. Another common convention is to append “-able” or “-ible” to the end of the name as in `Configurable` or `NamedValueAccessible`.

NamedValueAccessible

Let's use that last one as an example. The `NamedValueAccessible` mix-in adds the method `valueForKey()` to whatever class with which it is joined. For `obj.valueForKey(name)`, this method will return one of the following:

- `obj.name()`
- `obj._name()`
- `obj.name`
- `obj._name`

In other words, `valueForKey()` looks for methods or attributes, either public or private, in order to return a value for the given key. The design of this method reflects the fact that Python objects often provide information through both attributes and methods. See Listing 2 for the implementation.

Listing 2. A Mix-in for Uniform Value Access

A useful application of this mix-in is to implement generic code for writing logs (see Listing 3).

Listing 3. Applying the NamedValueAccessible Mix-in for Logging

By simply adding new keys to the `logColumns()` method, the log can be expanded without having to modify the code that generates it, which is found in `logEntry()`. More importantly, you can imagine that `logColumns()` could read its list of fields from a simple configuration file.

The transaction object itself is free to provide the given values via either methods or attributes, due to the flexibility of the `valueForKey()` method. Making mix-ins flexible increases their utility and is an art that can be developed over time.

Mixing It in after the Fact

So far we have seen examples of using mix-ins during the construction of classes. However, Python's dynamic nature also allows us to mix in functionality at runtime. The simplest technique for doing so is to modify the base classes of the given class, as described earlier. A function allows us to keep this operation opaque and enhance it later if need be:

```
def MixIn(pyClass, mixInClass):  
    pyClass.__bases__ += mixInClass
```

Let's consider a situation that makes the utility of `MixIn()` obvious. In the construction of internet applications, keeping domain classes separate from interface classes is generally a good idea. Domain classes represent the concepts, data and operations of a specific application. They are independent of operating system, user interface, database, etc. Some writers refer to domain objects as business objects, model objects or problem space objects.

Keeping the domain and interface separate makes sense for various reasons. An individual focus is created for two key areas that are largely independent: What is the subject material of the problem? And, how should that be presented? New interfaces can be constructed without modifying or rewriting the domain classes. In fact, multiple interfaces can be provided.

Domain classes for a story publishing system might include `Story`, `Author` and `Site`. These classes contain essential attributes (such as title, body, name, e-mail, etc.) and various operations (save, load, publish, etc.).

One interface for such a system could be a web site that allows users to create, edit, delete and publish these stories. When developing such a site, it would be useful if our domain classes, such as `Story`, have the methods `renderView()` and `renderForm()`, which write HTML for either displaying the story or editing it with a form.

Using mix-ins, we can develop such functionality outside of the domain classes:

```
class StoryInterface:
    def renderView(self):
        # write the HTML representation of the story
        pass
    def renderForm(self):
        # write the HTML form to edit the story
        pass
```

And within the code that backs the web site, mix it in like so:

```
from MixIn import MixIn
from Domain.Story import Story
MixIn(Story, StoryInterface)
```

If you decide to create a GUI interface for the publishing system, you don't have to take the HTML machinery with you (or vice versa). The domain classes focus on providing necessary data and operations, ensuring that when developing the GUI, you will have what you need.

One could argue that a new class might be created to bring the two together:

```
class StoryInterface:
    ...
from Domain.Story import Story
class Story(Story, StoryInterface): pass
```

Or one could argue that `StoryInterface` might be made a subclass of `Story` in order to achieve the same benefit. However, consider the case when `Story` already has other domain subclasses:

```
class Story: ...
class Editorial(Story): ...
class Feature(Story): ...
class Column(Story): ...
```

Existing subclasses of `Story` are in no way affected by simply creating a new `Story` class or subclass. But a dynamic mix-in for `Story` will also affect `Editorial`, `Feature` and `Column`. That is why many times the static approach does not work in practice, thereby making the dynamic approach not only clever, but necessary.

Also, consider the case where `Story` objects are created in parts of the code where `Story` is hard-coded. While poor practice, this is common. In this

situation, creating subclasses of Story will have no effect on the code that ignores them.

One warning regarding dynamic mix-ins: they can change the behavior of existing objects (because they change the classes of those objects). This could lead to unpredictable results, as most classes are not designed with that type of change in mind. The safe way to use dynamic mix-ins is to install them when the application first starts, before any objects are created.

Advanced Versions of MixIn()

The first enhancement we can add to MixIn() is to check that we're not mixing in the same class twice:

```
def MixIn(pyClass, mixInClass):
    if mixInClass not in pyClass.__bases__:
        pyClass.__bases__ += (mixInClass,)
```

In practice, I find more often than not, that I want my mix-in methods to take a high priority, even superseding inherited methods if needed. The next version of the function puts the mix-in class at the front of the sequence of base classes but allows you to override this behavior with an optional argument:

```
def MixIn(pyClass, mixInClass, makeLast=0):
    if mixInClass not in pyClass.__bases__:
        if makeLast:
            pyClass.__bases__ += (mixInClass,)
        else:
            pyClass.__bases__ = (mixInClass,) + pyClass.__bases__
```

To make Python invocations more readable, I suggest using keyword arguments for flags:

```
# not so readable:
MixIn(Story, StoryInterface, 1)
# much better:
MixIn(Story, StoryInterface, makeLast=1)
```

Listing 4. Our Final Version of MixIn

This new version still doesn't allow methods in the actual class to be overridden with methods in the mix-in. In order to accomplish that, the mix-in methods must actually be installed in the class. Fortunately, Python is dynamic enough to accomplish this. Listing 4 gives the source code for our final version of MixIn(). By default it will install the methods of the mix-in directly into the target class, even taking care to traverse the base classes of the mix-in. The invocation is the same:

```
MixIn(Story, StoryInterface)
```

An extra `makeAncestor=1` argument can be provided for the new `Mixin()` to get the old semantics (e.g., make the mix-in a base class of the target class). The ability to put the mix-in at the end of the base classes has been removed, since I have never needed this in practice.

An even more sophisticated version of this function could return (perhaps optionally) a list of methods that clash between the two, or raise an exception accompanied by such a list, if the overlap exists.

Installing Mix-ins Automatically

When making heavy use of after-the-fact mix-ins, invocations of the `Mixin()` function become repetitious. For example, a GUI application might have a mix-in for every domain class in existence, thereby requiring a call such as this for each one:

```
from Domain.User import User
Mixin(User, UserMixin)
```

One solution is to bind the mix-ins to their target classes by name and have the application install these at startup. For example, all mix-ins could be named directly after the class they modify and put into a `MixIns/` directory. The code in Listing 5 will install them.

Listing 5. Detecting and Installing Mix-ins Named after Their Classes

Additional Uses

While it's fun to explore more sophisticated versions of the `Mixin()` function, the most important key is the ability to apply them in order to improve your software. Here are some additional uses to stimulate your imagination:

- A class could augment itself with a mix-in after reading a configuration file. For example, a web server class could mix in `Threading` or `Forking` depending on how it's configured.
- A program could provide for plug-ins: software packages that are located and loaded at launch time to enhance the program. Those who implement plug-ins could make use of `Mixin()` to enhance core program classes.

Summary

Mix-ins are great for improving modularity and enhancing existing classes without having to get intimate with their source code. This in turn supports other design paradigms, like separation of domain and interface, dynamic configuration and plug-ins. Python's inherent support for multiple inheritance,

dynamic binding and dynamic changes to classes enables a very powerful technique. As you continue to write Python code, consider ways in which mixins can enhance your software.

Chuck Esterbrook is a consultant, writer and entrepreneur, using Python (<http://www.python.org/>) and Webware (<http://webware.sourceforge.net/>). He can be reached at ChuckEsterbrook@yahoo.com.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Managing Your Money with GnuCash

Robert Merkel

Issue #84, April 2001

GnuCash developer, Robert Merkel, explains the functionality of the Quicken alternative.

A refrain many Linux users and advocates hear again and again is “The only reason I keep Windows around is for Quicken.” Since 1997, the GnuCash Project (formerly known as xacc) has been working to provide an accounting solution for Linux. GnuCash 1.4, the current stable release, provides a simple, easy-to-use but powerful home accounting system. Development continues at a rapid rate, both through volunteers and employees of Gnumatic (<http://www.gnumatic.com/>), a company formed to support the continued development, marketing and productization of GnuCash and related projects.

Obtaining and Installing GnuCash

GnuCash is GPL software, so you can download the source code, as well as RPMs, from the GnuCash web site (<http://www.gnucash.org/>). Several distributions, including Mandrake, SuSE, Debian and Red Hat, also include GnuCash as part of their systems. While GnuCash requires the GNOME libraries, it will run quite happily with any window manager, including KDE. The current stable version is 1.4.9, but that may have changed by the time you read this. Versions 1.5.x are the next development series. These are intended for those who wish to get involved with where GnuCash is going, rather than its use in production environments.

Building GnuCash can be a little problematic for some people as it requires the installation of a couple of uncommon build tools. In addition to a working GNOME development environment, you need Perl, **guile**, **SWIG** (available from <http://www.swig.org/>) and **g-wrap**, another tool now maintained by Gnumatic and available from the Gnumatic web site. The dependency on SWIG is likely to go away sometime in the next development series (1.5.x). GnuCash uses the

standard GNU build environment, so compiling is just a matter of typing **./configure; make; make install**.

While GnuCash's primary development environment is Linux on x86, it has been successfully compiled and run under virtually all the architectures Linux supports, as well as FreeBSD/x86 and Solaris/x86. It should also run on just about any platform to which GNOME has been ported.

Remedial Accounting 101

While many other low-end accounting programs use a “cashbook” single-entry approach to accounting, GnuCash has always used a double-entry accounting system. While slightly more complex, double-entry accounting is a far more accurate accounting technique. You'll find that double-entry is a powerful way of organizing your finances. If you are already familiar with double-entry accounting, you probably can't imagine working any other way. While it's technically possible to use GnuCash for single-entry accounting, you'd be crazy not to take advantage of GnuCash's full abilities.

So, how does double-entry accounting work in GnuCash? Every time you spend, receive or transfer money is a “transaction”. When you write a check to pay your phone bill, that's a transaction. When your paycheck goes into your account—another transaction. When you go to the ATM to get some cash for a night out, that's a transaction. When you buy a burger with that cash...you guessed it, it's a transaction (whether it's practical to account for personal burger purchases is another question). In each case, money is transferred from one place to another, and the amount taken from one place must exactly equal the amount put somewhere else. Money never appears or disappears from thin air.

Similarly, in GnuCash every transaction involves money transferred from one account to another. “Huh? Two accounts?” I hear you saying. “Okay, for a bill payment one of those accounts might be a checking account, but what's the other?” Good question. The answer is that “accounts” can be much more than checking or credit card accounts. For instance, you can create an “expense account” for “Phone Bills”. Therefore, paying a phone bill would involve two entries, one in your checking account, and one in the “Phone Bill” expense account. These amounts must exactly match, something that GnuCash immediately checks. At this point, you might think that making two separate entries is a pain. GnuCash, however, lets you make both entries at the same time. More information on double-entry accounting is available in the GnuCash documentation (on-line at gnucash.org/docs/C/xacc-doubleentry.html).

So, let's create some accounts in GnuCash and record a \$100 phone bill payment. Start up GnuCash. If you installed from a binary package there's

probably a menu entry somewhere, but if that's not set up just type **gnucash** at a command-line prompt. You'll see the main window. First, we'll need to create some accounts. One way to do so is to click on the "New" button in the toolbar, which brings up a dialog box for creating accounts.

We'll create an account to represent our checking account. In the dialog box, enter the account's details. Most of the fields are self-explanatory, but there are a few that could use some explanation. The "Account Code" is used to determine the order of display in the main window and has no other effect on the account. If you don't care what order they are displayed in, you can safely leave it blank. "Account Type" is where things start to get interesting. GnuCash has 11 different account types. Our checking account is a "bank" account. Finally, you must select a "parent account". What's that? All will be revealed later, but for now just select "new top level one too".

Next, we'll create the "Phone Bill" expense account. Click on the "new" button, and repeat the process. The currency for the new account must be the same as for the bank account (transferring between different currencies is performed using special "currency" accounts). Make sure you set the type of the account as "expense". You should also select "new top level account" for this account.

If you're going to pay a phone bill, you've already got some money in your bank account. According to the rules of double-entry accounting, that money had to come from somewhere, right? Therefore, we create a special "equity" account. Repeat the process you used to create the previous accounts, but call this one "My Equity" and make it of type "equity".

Now, enter the initial bank account balance. Open the register for the newly created bank account by double-clicking on the entry in the main window. The register shows all the transactions and a running balance for the current account. At the bottom of each register there is always a blank line to enter new transactions. The column headings display transaction fields: date, number, description, transfer (which account the transaction involves transfer from/to), deposit, withdrawal and balance.

Let's say that your current balance is \$1,000.00, and we'd like to record that. The date is already set to today's date, so there's no need to modify that field. Use the tab key or the mouse to move to the next field, the description field, where we type **Opening Balance**. Next, we select the account where we're transferring money from—the special "My Equity" account. Enter the amount (\$1,000.00) into the deposit field, and press enter to record the transaction. A new blank transaction will be created, and the balance will change to record the current account balance.

Finally, enter the phone bill. Enter the appropriate information in each field: check number, description of the transaction and the other account involved in the transaction. The transaction is recorded, the new balance for the account is calculated and a new blank transaction is recorded.

Creating an Account Hierarchy

The simple example has covered most of what's needed to use GnuCash to track your expenses. You can go ahead and create expense, income, bank and cash accounts to meet your needs. However, if you conduct a lot of different types of transactions and end up with a bunch of accounts, there are going to be a lot of accounts for fairly similar types of transactions. Wouldn't it be convenient if you could somehow group these accounts? GnuCash lets you do just that. In fact, GnuCash is designed around the concept of a chart of accounts or account hierarchy, in which accounts are placed in a tree structure.

For instance, wouldn't it make sense to group together all the different sorts of utility bills, such as electricity, water and phone? GnuCash can do that. First, create a general Utilities expense account, just like you created the Phone Bill expense account. Next, create an Electricity expense account, but this time, in the "parent account" field of the dialog box, select Utilities rather than "New top level account". Repeat the process to create a Water expense account under the Utilities account.

And now let's just add an account for phone bills...uh, wait, we already have one of those. What do we do now? Do we delete that account, make a new one and re-enter all those transactions? No, we can move the "Phone Bill" account to make it a child of the Utilities account. In the main window, click on the Phone Bill account, then either click on the "edit" toolbar button or right click on the highlighted main window entry, and select "edit account". Either way, you bring up the "edit account" dialog box where you can change any properties of the selected account. Change the parent account to Utilities, and click "OK". That's it!

Importing from Quicken

So far, we've assumed that we've never used accounting software before. For many people that's not the case, and Quicken's QIF format is probably the most common way that people export data from their systems. Microsoft Money can also export QIF. Check the documentation of your particular system to find out whether your package can export QIF. Unfortunately, Quicken's QIF file has some rather interesting "features" and design decisions (including recording dates differently depending on your location), so importing your data into GnuCash requires a little manual intervention along the way.

GnuCash's on-line help system (accessible from the help menu) covers the importation process in detail. In fact, the on-line help system is a comprehensive guide to all of GnuCash's features and gives quite a lot of accounting background, not just the details of GnuCash's operation. To use the on-line help just select "help" from the main window. Many dialog boxes also have a help button available.

In any case, let's assume you have exported a QIF file with all your accounts, that you want to import to GnuCash. From the GnuCash main window, go to the File menu and select "import QIF", which brings up the import dialog box. First, select the QIF file you wish to import, and select the default currency for this QIF file. Next, give a name to the default account, and specify the currency in which this QIF records transactions. Then, click on "load file". If you have multiple QIF files you wish to import at once, repeat the process.

The QIF importer then examines the QIF files, creates an account structure that matches the original QIF categories and accounts with GnuCash accounts, and checks for any duplicate transactions. This is quite a complex process so don't be concerned if it takes a while and GnuCash doesn't respond to user input. It also guesses the appropriate account type for each category. While the QIF importer usually guesses correctly, it's wise to check. You can also add descriptions and change the parent of accounts. If you want to do this later, you can do it from the main window. Anyway, when you've checked everything just click OK, and importation will be complete!

You can also import QIF into existing GnuCash files. Use the same process as above, but this time you'll probably have to use the "accounts" and "categories" tabs of the QIF importer to tell GnuCash in which of your existing accounts to place various transaction categories. These days, many banks are able to provide QIF files representing all transactions in an account over a period of time. You can use the QIF importer to import those transactions automatically into the system.

Reconciliation

GnuCash can also assist you in reconciling your records against your bank statement. When you receive a bank statement, just choose that account and then right click and select "reconcile". Tell GnuCash the ending balance on the statement, and the reconcile window is displayed (see Figure 1).

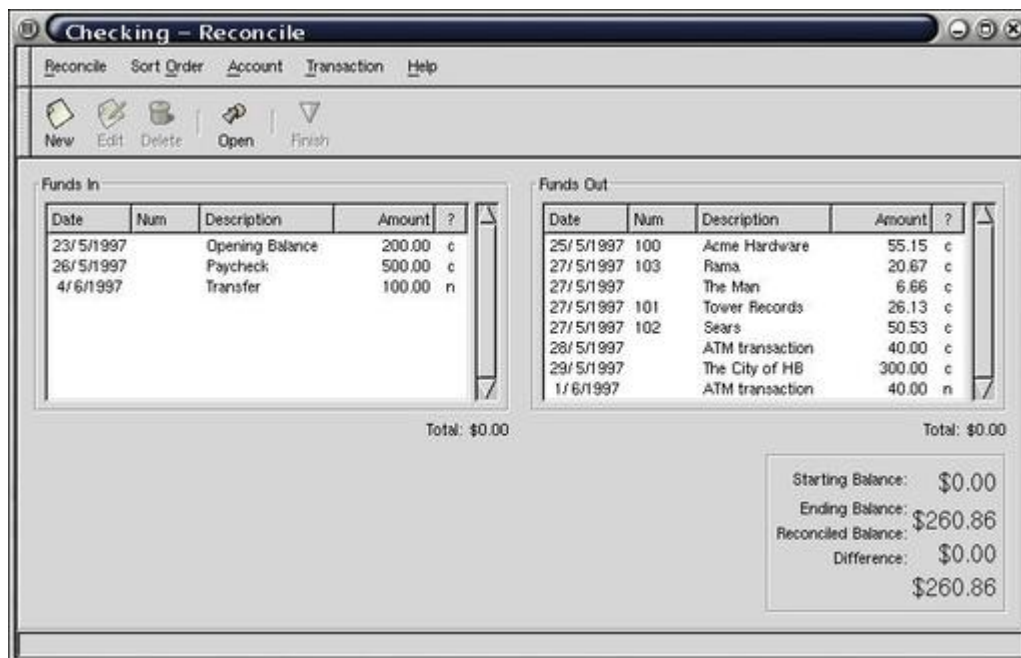


Figure 1. The Reconcile Window automates the process of reconciling your records against your bank's.

You should then go through your bank statement, and check the uncleared transactions against the transactions on the statement. If they match (and the amounts are correct) click that transaction to clear it. When the process is complete, the difference field at the bottom of the window should be \$0.00. If it isn't, there's a discrepancy between your record and the bank's. Either you've missed a transaction (have you checked the bank's fees and government charges?), an amount is wrong or the bank has made an error (possible, but unlikely).

Stocks and Other Investments

GnuCash has extensive facilities for tracking investments in stocks and mutual funds. When you purchase a new stock or open a mutual fund, you should open a stock or a mutual fund account, just like a bank account. When you open the account, make sure you enter the correct ticker symbol in the Security field, and select a source quote.

While automated, updating prices is done from outside GnuCash with the **gnc-prices** Perl script. When the script is run, all the available prices are updated.

Recording stock purchases and sales is performed using the stock account register, which works just like the other registers except you should enter a price and a quantity of shares/units instead of just one value.

What? You want to analyze the performance of your shares? For that, you need...

Reports

GnuCash has some very useful reporting capabilities that you can view directly or export as HTML for permanent storage or printing. Current CVS has direct printing support through gnome-print, but this has not made it into release versions yet.

One of the most basic reports is the “balance sheet”. This report summarizes your assets and liabilities, thus showing your net worth. To display the report, simply select “Balance Sheet” from the report menu. A report window will open and display the report. By default, it displays the balance sheet as of this moment, but that's adjustable. Click on the “parameters” button, which brings up a dialog box from which you can change options for the report.

An overview of your stock portfolio is also available, as well as the account balance tracker, a very useful report for tracking the growth of a specific stock or mutual fund. This report also supports graphing if you have **gnuplot** installed. Finally, the transaction report lists all transactions meeting a set of criteria. This flexible report is often useful for extracting specific information not available in other reports (see Figure 2).



Date	Num	Description	Memo	Account	Amount
May					
23/ 5/1997		Opening Balance		Checking	200.00
25/ 5/1997	100	Acme Hardware	building supplies	Checking	(55.15)
26/ 5/1997		Paycheck		Checking	500.00
27/ 5/1997		The Man		Checking	(6.66)
27/ 5/1997	101	Tower Records	cool CDs	Checking	(26.13)
27/ 5/1997	102	Sears	power tools	Checking	(50.53)
27/ 5/1997	103	Rama	Mmm... Beer	Checking	(20.67)
28/ 5/1997		ATM transaction		Checking	(40.00)
29/ 5/1997		The City of HB	ball	Checking	(300.00)
					200.86
June					
1/ 6/1997		ATM transaction		Checking	(40.00)
4/ 6/1997		Transfer		Checking	100.00
4/ 6/1997		Opening Balance		Savings	600.00
4/ 6/1997		Transfer	[To: Checking]	Savings	(100.00)

Figure 2. The Transaction report can be used to display and subtotal transactions in a variety of ways.

If the current reports do not meet your needs, you can write your own. The “hello world” report is the GnuCash authors' test bed and serves as a basis for those who wish to write a custom report.

Future Plans

While GnuCash is stable and useful right now, Gnumatic and the rest of the GnuCash community have big goals for the future. Already, the development CVS tree has several new features, and many more are planned.

The coming 2.0 release is under heavy development, and plans are afoot for a Q1 2001 release. Some of the features that will be part of 2.0 include:

- A new, XML-based file format that will be more compact and versatile.
- Printable reports (courtesy of gnome-print and the gtkhtml widget).
- Better graph capabilities through GUPPI (which will hopefully be widely adopted as the standard Gnome graph infrastructure). Development of GUPPI has been supported by Gnumatic.
- Much improved report flexibility and customizability.
- Much improved stock market reporting.

Further improvements to GnuCash's personal accounting abilities are planned in the long term. Some of these include:

- Improvements to on-line stock quote gathering and currency quotes.
- Interfacing to on-line banking, including OFX and other standards.
- Interfacing to Palm PDAs.
- Full documentation of the guile API to make writing extensions easy.

While GnuCash's current emphasis is on personal accounting, the large number of requests for business accounting functionality we receive on the mailing list are not being ignored. GnuCash will form the basis of a full-featured, multiuser accounting package with a database engine; CORBA bindings; and support for payroll, inventory, invoicing and the other features that larger businesses require from accounting systems. Gnumatic will be in the forefront of this, offering its services to customize and support GnuCash for specific customers, industries and countries.



Robert Merkel started hacking on GnuCash because he was sick of supporting his father's Windows box. Robert is now a full time GnuCash developer and an employee of Gnumatic, Inc.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Enterprise-Level Health-Care Applications

Gary Bennett

Issue #84, April 2001

eTransMan saves the day.

Today most health-care companies are struggling to find ways to improve patient care, reduce costs and provide more and better service. They increasingly see technology as a solution to these requirements. Our company is no different. We have approximately 1,300 employees in 13 states and 40 offices. Our existing patient care system and financial accounting system were developed over ten years ago and were not designed to handle the requirements of a large, distributed organization.

Many members of our executive team have backgrounds in high-volume claims processing, where millions of claims have to be processed daily on high availability systems. Our CEO and the executive team defined the following requirements for our new IT applications:

- It must be scalable, portable, reliable and secure.
- It must address the requirements of the Health Insurance Portability and Accountability Act (HIPAA).
- It must integrate all our applications and provide business-to-business (B2B) capability.
- It must meet the “four As” of availability: accessible, anytime, anywhere and on any device.
- It must be cost effective.

Enterprise-Level Application

Scalability is a matter of survival. Many vendors have said, “hardware is so cheap, just throw more hardware at it.” This is a good solution until you are responsible for a budget and getting it past a CFO. In our situation we must not sacrifice one byte of memory, one processor cycle nor one block of disk to bloated, inefficient applications and systems.

Portability and scalability are related. If our application is written on an Intel platform, but must run on a mainframe to perform as needed, that application must be portable to different operating systems running on different hardware. Our hardware selection today may change a year from now based on speed, cost, support, reliability and available talent to run the system. We can't replace our applications whenever we change our hardware.

HIPAA, passed by the US Congress and signed into law in 2000, has tremendous implications for health-care companies. These laws take effect in 2002. HIPAA's authors intended to standardize how claims are processed, how data is exchanged between companies and how patient information is accessed and stored. The applications and systems we run must conform to HIPAA requirements and must change with the regulatory environment. Health-care companies will spend two to three times the cost of Y2K on HIPAA conformance, according to industry analysts.

Industry analysts also claim that health care is one of the last industries to take significant advantage of advances in information technology. Many health-care companies' information systems run on old, sometimes very old, technology, and our systems must communicate with them. When we enroll a patient in our medical information system, we must automatically enroll that patient in our pharmacy provider's system so the patient can fill their prescriptions anywhere in the US the same day. We must also send information to our medical supply company to allow the patient to get medical supplies without excessive delay.

Finding the Solution

Our company evaluated over 50 health-care applications. None could address our needs. Some companies had just invested millions of dollars to go from DOS application to client/server, 32-bit Windows applications. Some had DOS applications and just looked at us funny when we tried to explain what we wanted. In the end we found nothing that fit our needs. Our solution was to put together a team of highly experienced developers with backgrounds in high-volume transaction processing and build a transaction backbone and the applications to run on top of it.

Architecture Decisions

Our first task was to define the broad outlines of the architecture. We wanted an N-tier architecture, but beyond that we wanted one that placed no limits on possible solutions. We evaluated several transaction engines and application servers. All failed to satisfy our requirements. Commercial transaction engines were either extremely expensive to purchase, maintain and develop on or they were too inflexible or unreliable for us.

How much reliability do we require? If an e-tailer goes down .1% of the time, about nine hours per year, the worst likely outcome is the loss of some orders. However, when a nurse needs to access patient records to learn a patient's medication and dosage, downtime is a different issue. It's simply not acceptable.

eTransMan

The build or buy decision was easy, there was nothing to buy. We chose to build a transaction engine on steroids—**eTransMan**. eTransMan, eCommerce Transaction Manager, is the result (see Figure 1).

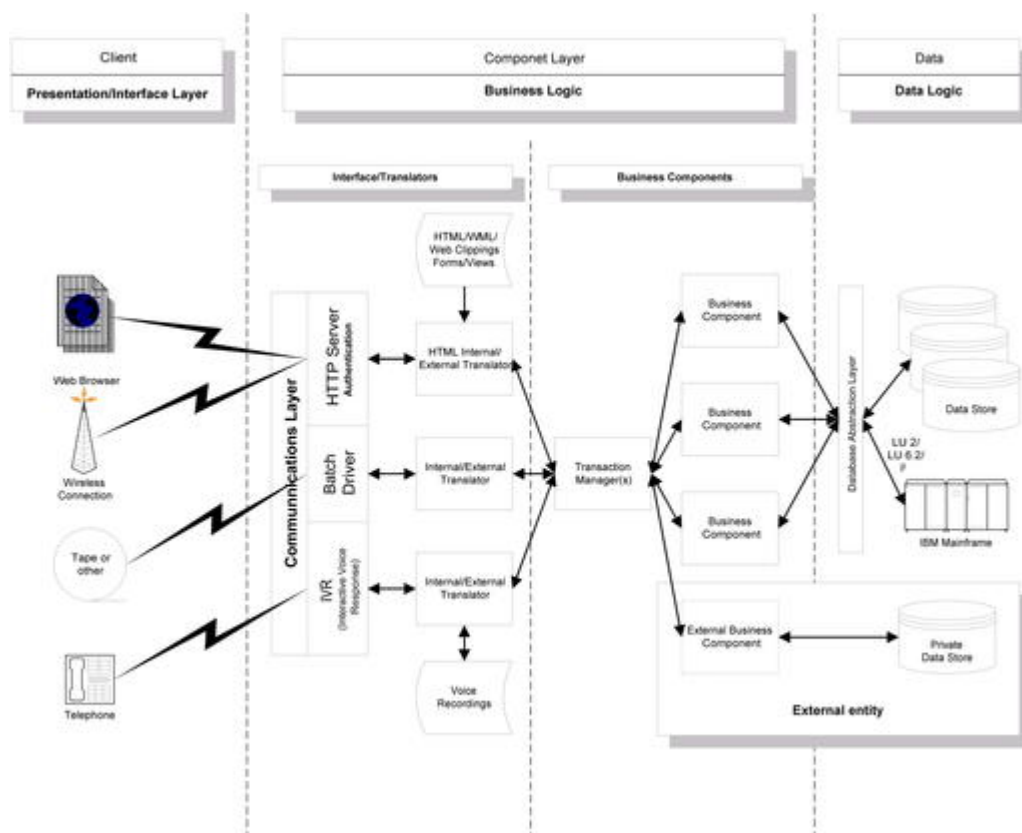


Figure 1. Architectural Overview

eTransMan is the heart of an easy-to-use architecture that provides an environment where developers easily build and manage N-tier applications for health care's pervasive computing environment. Developers write their business processes as modular components using any computer language—C, C++, Java, COBOL, Perl, Visual Basic and even assembler. eTransMan runs these components in a highly scalable, portable, high-performance, secure transaction environment.

Developing components for eTransMan is simple by design. Our company's developers write business components in C++ and Java. Other companies may not have developers with these skills. They may only have COBOL, Visual Basic or PowerBuilder developers. With eTransMan, companies avoid the time and

expense of a weeklong class just to learn how to the transaction engine followed by another week of Java or C++ training.

eTransMan's Architecture

The Transaction Manager is the architectural hub of eTransMan. The Transaction Manager provides fault tolerance, load balancing, scheduling, security and business component location (routing).

eTransMan's architecture is based upon the classic N-tier model with some enhancements. This model separates the presentation, business logic and data access layers. The main goals of this architecture were to:

- create a transaction manager capable of very high transaction throughput on modest hardware;
- create a component-based architecture that is robust, reliable and easy to manage and control in a production environment; and
- enable component developers to design and add business components to the infrastructure as simply and as quickly as possible, with a minimum understanding of the architecture.

The Transaction Manager is written in C/C++ on Linux and compiled with the open-source gcc compiler. We can thus easily port to any system that supports gcc, from embedded platforms to an IBM 390 mainframe. eTransMan has exceeded performance expectation due to the efficient engineering provided on this platform. We are currently running eTransMan on two dual PIII 700 servers. The total production hardware cost was under \$30,000 for 40 sites serving 450 on-line users (see Figure 2). Resources are only allocated when needed, enabling rapid responses on our modest hardware. Load average is never more than 2%. eTransMan is a compiled application, so it runs without an interpreter.



Figure 2. Attached File Servers

eTransMan includes many features that support high-application availability: processing of availability checks, timeout checks, automatic component restart and recovery procedures and user-definable recovery procedures. Business components run as dæmons for extremely fast response from the Apache web server. eTransMan not only controls the flow of activity in the application but also ensures smooth, efficient operations of all components (see Figure 3).

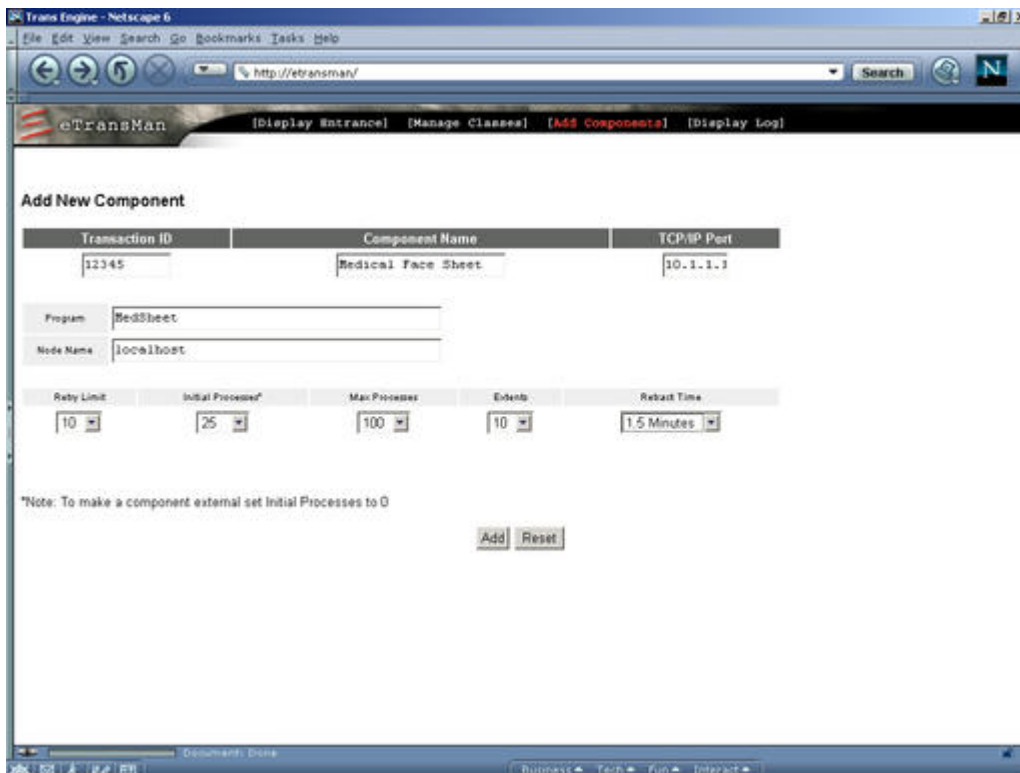


Figure 3. Managing the Flow of Activity

eTransMan balances the load between multiple identical components to ensure maximum application throughput. eTransMan tracks which components are available for a specific transaction. It dynamically creates additional component instances to handle increased transactional demands, destroying them when they are no longer needed. This feature allows eTransMan to allocate resources optimally to satisfy current demands. Using Linux's open architecture, eTransMan can dynamically map requests to components, exercising broad control over the flow of processing for the applications.

By multiplexing and managing the relationships between clients and applications, eTransMan provides all the capabilities needed for mission- and business-critical applications, including the capacity for large volumes of centralized or distributed data, high-message throughput, high data integrity and security and high-application availability, including 24/7 processing.

Resources are used as specified by configuration files but are only used as necessary, enabling rapid responses on modest hardware. We built a web

interface to manage our configuration files. The eTransMan architecture follows the N-tier model. The workflow progresses as follows:

- Intelligent data clients, such as Linux workstations, Windows 2000/98/95 workstations, Mac workstations, Wireless Devices, Interactive Voice Response Systems (IVR), virtually all commercial versions of UNIX, batch inputs, point-of-sale terminals, video data stream and other devices optimized for specific data and presentation functions gather information for subsequent processing and display information.
- Client-specific interpreters prepare data for eTransMan. Interpreters package the incoming data stream in a format that the business components expect. The interpreters then forward the data client's request to eTransMan. Adding a new user client is as easy as creating a translator that can convert client-specific data formats to and from the simple eTransMan data exchange format. The business logic never changes.
- eTransMan validates the transaction request and then forwards the request and data to the business component.
- Components can access databases with several methods. To minimize cost and increase efficiency we developed our own pooled database component to access our Oracle 8i database running on Linux. We can connect to other databases via native libraries, ODBC, JDBC or via an HLLAPI component on a mainframe. Our business component returns the response and associated data back to eTransMan, which sends the response back to the interpreter. The interpreter formats the data according to its own rules.
- One type of data interpreter supplied with the eTransMan architecture matches the response and associated data with a template. The interpreter merges the data with the template in the client's native format and returns the result. Templates can handle singleton or tabular data in the same response.

The Presentation Layer

When the Transaction Manager receives a response from the component, it forwards the response to the communication layer that sent the request. The communication layer then translates the result into a format that is recognizable by the data client. The data is then sent back to the client in its native format.

We needed to make changing the user interface as easy as possible. The first test came when we wanted to adapt an existing PC-based browser interface to a wireless Palm Pilot and cell phone. We wanted to allow our field personnel to use a very simple tool—cell phones—to access the database. Our first challenge

was to deliver driving directions to a patient's home on a browser-enabled cell phone or wireless Palm Pilot. By using our translator technology, we were able to take an existing web-based application (in this case a report with 150 columns of data) and adapt it using a new template to the wireless world. We ported to a WML interface in less than four hours, with no recoding of the business or database components (see Figure 4). We reproduced this feat using web clipping technology to the Palm VIIx. It took four hours because we were not familiar with WML or web clipping. The Yankee Group says there are up to 12 million potential users in the medical industry of wireless technology. The Web is our tool today, but we will soon witness an explosion of wireless interfaces. We would be shortsighted to have an architecture that would require any recoding of business components (see Figure 5).



Figure 4. Wireless Driving Directions

The screenshot shows a web browser window titled 'VistaCare: Patient Enrollment Application - Netscape 6'. The address bar shows 'http://co...'. The main content area displays a 'Detail Patient Form' with the VistaCare logo on the left. The form includes a navigation menu with options like 'Home', 'Records', 'Search', 'New', and 'Logout'. The patient information section is titled 'PATIENT INFORMATION' and includes fields for Last Name (SCHULTZ), First Name (ROBERT), MI (D), Home Phone ((480) 961-0092), Social Security # (535-55-5555), Address (7306 N. 90th Pl, BFT 215), City (PHOENIX), State (AZ), Zip (59199), County (MARICOPA), MSA (11029), and Zip (21:22:Y3). There are also buttons for 'Face Sheet' and 'Certification'.

Figure 5. Patient Form

The Communication Layer

Interfaces and translators, which are part of the Communication Layer, package data for the business components. Interfaces and translators also format data that is sent back to the original client source. As one technique to format the data, the interpreter can use a template; for example, an HTML template could represent a classic master detail report.

Interfaces and translators enable developers to focus on the business logic without the distractions of how the data is formatted or presented. These translators can use a common stream of data coming from the business component and move it to virtually any front end: Web, IVR, wireless or even print streams.

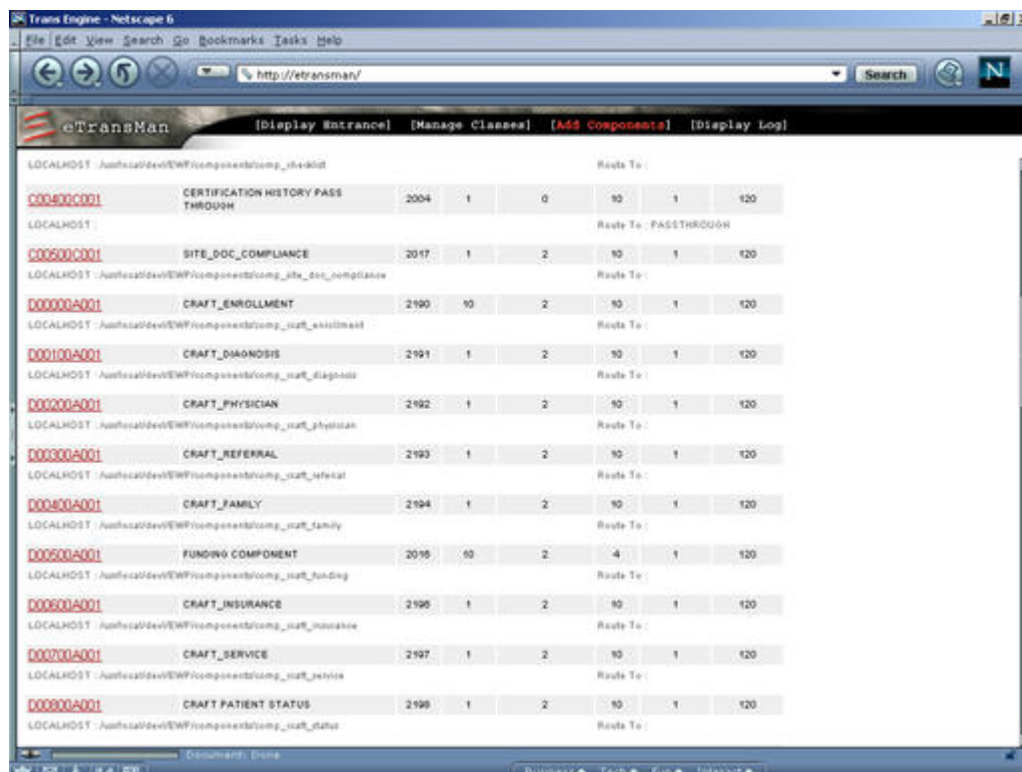
More importantly, this allows reuse of components if the data client changes. Having the presentation and communication logic separate from the business logic speeds software development and allows for faster application deployment and updates. It also enables developers, particularly web and graphics programmers, to focus on the presentation without having to worry about the business logic or database I/O.

The eTransMan Business Component Software Model

eTransMan uses a component software model. Data clients and other requesters invoke managed business components and developers design business components or modify their components by focusing on the business processes. This development strategy enables developers to build components quickly with their existing knowledge and skills.

We needed the ability to alter the availability of individual business components without affecting any other components. We found that this feature was missing from many of the commercial and open-source transaction platforms that we surveyed.

Health care's computing environments require administrative vigilance. Performance and security monitoring, failure and alert management, and a host of other operational tasks are more difficult in a distributed environment. We also designed web-based administration capabilities to bring eCommerce applications on-line and off-line. The easiest way to address this was to use the interface and translation layer to manage it through the Web. We use eTransMan to manage eTransMan (see Figure 6).



ID	Component Name	Year	Column 4	Column 5	Column 6	Column 7	Column 8	Column 9
00490A001	CERTIFICATION HISTORY PASS THROUGH	2004	1	0	90	1	120	
00500A001	SITE_DOC_COMPLIANCE	2017	1	2	90	1	120	
00000A001	CRAFT_ENROLLMENT	2190	90	2	90	1	120	
00100A001	CRAFT_DIAGNOSIS	2191	1	2	90	1	120	
00200A001	CRAFT_PHYSICIAN	2192	1	2	90	1	120	
00300A001	CRAFT_REFERRAL	2193	1	2	90	1	120	
00400A001	CRAFT_FAMILY	2194	1	2	90	1	120	
00500A001	FUNDING COMPONENT	2015	90	2	4	1	120	
00600A001	CRAFT_INSURANCE	2196	1	2	90	1	120	
00700A001	CRAFT_SERVICE	2197	1	2	90	1	120	
00800A001	CRAFT PATIENT STATUS	2198	1	2	90	1	120	

Figure 6. etransman_manage

Data Abstraction Layer

eTransMan's Database Abstraction Layer provides a common interface for database interaction. This layer enables developers to interact with any database without modifying business component code.

The advantage of the Data Abstraction Layer is that a business component need not notice if the underlying data source changes, say from DB2 running on an IBM mainframe to Oracle running on Linux. Component programmers need not have their logic depend upon the specifics of a particular database. There are no SQL or HLLAPI manipulation routines in a component, for example. All of the data source access is through a specific component that translates generic requests into a format for a particular data source.

Unlike some transaction platforms that interact with a database using JDBC or ODBC, eTransMan can take advantage of the database's native libraries for database interaction. This engineering achieves fast database access with minimal overhead.

A metadata layer, stored and read by the component at instantiation, controls calling from the component to data access procedures for the database. This lets each software layer do what it does best: the database to read and write data, the application program to run business logic. This layered abstraction has already allowed us to move an application developed in one RDBMS to a different data layer.

Database Connection Pooling

Database Connection Pooling is a technique that minimizes the number of connections to a database. Database connections require intensive use of database and system resources. Frequent connection and disconnection leads to overall performance degradation. In high-transaction environments, connection pooling ensures that resources are used efficiently to optimize response times and data throughput.

Today, many database vendors license their products by the number of concurrent users, connections and/or number of power units based on MHz per processor on which the database is running. eTransMan helps maintain a low cost of ownership for their transaction platform by using only those system resources that are absolutely necessary to do the job.

If we were to connect our typical daily load of 200 on-line, we would need quite a fast machine full of RAM. However, we can support those 200 users with five pooled database connections, allowing eTransMan to predictively start more connections if required.

Summary

Our system provides high uptime. By building everything in redundant, restartable small components we can provide multiple paths for the application. Business components can be run on several servers if software redundancy is inadequate, providing reliability from even total server failure. eTransMan marks those components as unavailable and routes the business to another available server. Web servers already run this way, now the applications can, too.

Application requirements come and go. Businesses change, merge, get bought, get sold—nothing stays the same. We know we'll be running different business and database components in the future than the ones we're using today. With the right architecture we also know that we can make moves such as from Web to wireless without impacting the business components. The main lesson of this process is not to get locked in to a vendor, a technology, a predefined interface, a web server or to a database.

Gary Bennett (linuxrules@vista-care.com) is the director of IT for VistaCare, a hospice health-care company. He has over 17 years experience developing software in the medical, utility industry and military fields. When he is not looking over project Gantt charts he is looking over topo maps and planning backpacking trips in and around Arizona.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Server-Side Java with Jakarta-Tomcat

Reuven M. Lerner

Issue #84, April 2001

Simple ways to build web applications using servlets.

When I began to write server-side web applications, there were two mainstream choices: if you wanted the program to execute quickly, you chose C, and if you wanted to write the program quickly, you used Perl. C, as we all know, is great when the binary needs to be small, fast and efficient. But C's lack of automatic memory management and decent string handling, along with the extreme care that programmers need in order to use it, made it a poor second choice when compared with Perl.

But in the last few years, a number of programming languages have begun to challenge Perl for the server-side web programming mantle. In particular, Python has gained significant ground, thanks in no small part to the growth of the Zope web development environment.

But perhaps the largest groundswell of server-side programming is coming from the Java community. Java, as many of us might remember, began life as a client-side programming language. For the most part, applets are an unpleasant memory of what happens when you try to mix two client-side paradigms—a lesson that the increasingly common use of Flash seems to ignore.

The basic unit of server-side Java is the **servlet**, a small program that is executed in response to an HTTP request and that generates a legal HTTP response. Since servlets are written in Java, they are written as object classes, inherit from a servlet ancestor and can take advantage of Java's threading and exception handling. Moreover, servlets (like all Java programs) run inside of a Java virtual machine (JVM), an abstraction layer that can run on any operating platform. This means that the same servlets can run on nearly any operating system, providing even greater portability than CGI programs.

I have used servlets in a small number of projects so far, but this number is rising rapidly. Java is now the “in” language. This is partly because Sun has invested a large amount in its marketing, partly because it offers some technological and infrastructural advantages over its rivals and also because it poses a serious platform challenge to Windows. In addition, server-side Java is the cornerstone for an increasing number of Java-based application servers.

This month, we will begin to explore Java as a server-side programming language. As a beginning step, we will install the Jakarta-Tomcat environment for running servlets, as well as the associated Jasper environment for creating Java Server Pages (JSPs). In coming months, we will look at how to connect our servlets and JSPs to a relational database, as well as how to use Enterprise JavaBeans and the Enhydra application server for an even more powerful environment.

Installing Java

When I first started to work with Java on Linux a few years ago, the situation seemed fairly grim: while Linux was the best known, open-source operating system, and Sun was promoting Java as a universal programming language, it was difficult to impossible to get a good version of Java for Linux. Some volunteer porting efforts, particularly the one done by the Blackdown porters, were impressive, but installation was prone to problems and not nearly as stable as developers might have liked.

As I write this article in January 2001, the situation has changed dramatically: you can now download a Linux version of the latest Java development kit (JDK) directly from Sun's web site. Further, the Tomcat servlet/JSP system works just fine on Linux. As Linux picks up more steam, it is becoming an increasingly attractive platform on which to program in Java.

Because my main Linux box runs Red Hat 6.2, I downloaded the JDK 1.3 RPM from Sun's web site, <http://java.sun.com/>. In order to download the JDK, I had to sign up as a member of the “Java Developer Connection”; while I'm not thrilled by the notion of having to register in order to download software, it does not seem like a terrible price to pay. The RPM cannot be installed directly; first, you must agree to Sun's Java licensing agreement.

Once you have accepted the agreement, the RPM is unpacked and made available for install. You can then log in as root and install the JDK, which is placed in the `/usr/java` directory. By putting `/usr/java/jdk1.3/bin/` in your PATH environment variable, you can execute the **javac** compiler and the java runtime environment without having to specify an explicit path.

Once you have installed the JDK, you should run at least one simple test to ensure that it works. Listing 1 contains a simple program that can be invoked without any parameters and prints “hello, world” to STDOUT. If the program is passed through any parameters, it prints those parameters separated by a pipe character (|).

Listing 1. Test.java

To compile our test class (Test.java) into bytecodes (Test.class), use the Java compiler, javac:

```
javac Test.java
```

To run the program, we must invoke the Java runtime environment (java), giving it the name of our class (without the .class suffix):

```
java Test
```

If we don't pass any arguments, we get the following output:

```
Hello, world
```

We can, however, pass arguments to our program:

```
java Test a b "q r s" 123
```

In this case, we get the following output:

```
a|b|q r s|123
```

In addition to setting your PATH correctly, you should set the environment variable JAVA_HOME to point to the location of the JDK. If you use **bash**, you can simply put the following line inside one of your startup files:

```
export JAVA_HOME=/usr/java/jdk1.3
```

Installing Jakarta-Tomcat

Now that we have installed the JDK, we can install Jakarta-Tomcat. Jakarta is the overall name for Java-related projects sponsored by the Apache Software Foundation, and Tomcat is the ASF's project for servlets and JSPs. (JSPs, as we will see next month, are simply an easy way to create servlets.) Tomcat is meant to be the reference standard for servlets and JSPs on a variety of platforms, making it portable and easy to use Java in server-side web applications.

Unlike a CGI program, which executes within its own UNIX process, and unlike a mod_perl handler, which executes as a subroutine within Apache, servlets execute within a Java virtual machine. This JVM is known as a “servlet container” and can be the server itself (if the server is written in Java), embedded inside of the server or external to the server.

This article assumes that you will be using Apache, in which case the servlet container is external to the HTTP server. However, Tomcat is itself a full-fledged HTTP server, meaning that we can conduct some initial tests without having to configure Apache at all.

You can download and install the latest version of Tomcat from the Jakarta web site at <http://jakarta.apache.org/>. The Jakarta Project distributes software for a variety of platforms and on a number of schedules, in both source and binary formats. It may take a bit of looking, but you should be able to find a downloadable binary of the latest stable Tomcat release for Linux. As of this writing, the most recent stable version of Tomcat is 3.2.1, which I downloaded in the file jakarta-tomcat-3.2.1.tar.gz.

Once downloaded onto your computer, change to the directory where you want to install Tomcat, and open it up:

```
cd /usr/java
tar -zxvf jakarta-tomcat-3.2.1.tar.gz
```

Your /usr/java directory will now contain two subdirectories, one named jdk1.3 and the other jakarta-tomcat-3.2.1.

Just as you had to set JAVA_HOME to indicate where your Java distribution is located, you must also set the TOMCAT_HOME variable to indicate where Tomcat was installed. Those using bash can add the following line to one of their startup files:

```
export TOMCAT_HOME=/usr/java/jakarta-tomcat-3.2.1
```

If you are planning to write your own servlets, you will also need to tell Java where to look for the servlet-related classes. These are located in a Java archive (.jar) file, \$TOMCAT_HOME/lib/servlet.jar. If you use bash and don't otherwise set your CLASSPATH, you can set it as follows:

```
export CLASSPATH=$TOMCAT_HOME/lib/servlet.jar:.
```

You don't need to modify CLASSPATH in this way if you are only planning to run servlets that other people have written. The runtime Java servlet engine knows where to look for the appropriate .jar files, and its CLASSPATH is set correctly when you install Tomcat.

Once you have performed all of these steps, Tomcat is ready to go. You can start it up using the shell script under \$TOMCAT_HOME/bin:

```
$TOMCAT_HOME/bin/startup.sh
```

A number of diagnostic messages will appear on the screen. However, the main `servlet.log` log file is normally in `$TOMCAT_HOME/logs`.

You can check to see if Tomcat works by pointing your browser at port 8080 (the default) on the computer where it has been started. In other words, `http://localhost:8080/` should give you a welcome message, indicating that “this is the Tomcat default home page” with some additional links to examples of servlets and JSPs installed on the system. The example servlets should execute correctly, providing you with a demonstration of some simple tasks that we can perform with Tomcat.

Servlet classes are normally installed under a directory named `WEB-INF`, underneath the directory named in the URL; that is, the example servlet `RequestInfoExample`, which comes with Tomcat, is available at `http://localhost:8080/examples/servlet/RequestInfoExample`.

The actual Java `.class` file (as well as the `.java` source file for that class) is stored in `$TOMCAT_HOME/webapps/examples/WEB-INF/classes/RequestInfoExample.class`.

We will soon see how to configure additional directories for servlets. However, we will always have to install our classes under the directory `WEB-INF/classes`, and the `WEB-INF` hierarchy will be hidden from public view.

Listing 2. HelloWorld.java, a simple applet that handles the GET request method.

A Simple Servlet

We can test our Tomcat installation by placing a simple servlet, `HelloWorld.java` (see Listing 2), inside of the directory mentioned above, `$TOMCAT_HOME/webapps/examples/WEB-INF/classes/`.

Remember that Java requires filenames to match the class names. If you want to change the filename to `ABC.java`, you will have to change the class declaration inside of the source code to the same name. Otherwise, the Java compiler will complain with a fatal error.

To compile `HelloWorld.java` into an executable servlet, use the Java compiler, just as we would normally do:

```
javac HelloWorld.java
```

If your `CLASSPATH` environment variable was not set correctly, `javac` will complain that it cannot resolve the symbols `HttpServletResponse`,

ServletException and a number of other classes. Rectify this by setting your CLASSPATH to include servlet.jar, as indicated above.

Once the servlet has been compiled, you should be able to invoke it with `http://localhost:8080/examples/servlet/RequestInfoExample`.

If you attach a `firstname` parameter to the URL, the servlet should print your first name as well: `http://localhost:8080/examples/servlet/RequestInfoExample?firstname=Reuven`.

As you can see, this servlet is extremely simple. It imports a number of other useful Java packages, including the all-important `javax.servlet.*` and `javax.servlet.http.*` hierarchies. We then define our servlet as a subclass of `HttpServlet`. In doing so, we inherit all of the logic of `HttpServlet`.

Our `HelloWorld` servlet is particularly simple and includes a single method, **doGet**. `doGet` is invoked whenever the servlet is invoked with the GET method. HTTP supports a number of methods, but the most common are GET and POST; GET is typically used when a user directly requests a URL or clicks on a hyperlink, while POST is used when someone clicks on a “submit” button at the bottom of an HTML form. Because our servlet defines a `doGet` method but no `doPost` method, it can only handle GET requests.

The two arguments to `doGet` describe the HTTP request and response. If we want to retrieve information from the HTTP request, we use a method on our request object. For example, we can retrieve the value associated with the `firstname` parameter with the `getParameter` method:

```
String firstname = request.getParameter("firstname");
```

If no `firstname` parameter was passed in the request, the variable, “`firstname`”, will be assigned the null value. (This is distinct from the empty string, which indicates that the parameter was passed in the HTTP request but contained no value.)

We can similarly affect the HTTP response by invoking methods on our response object. For example, we can set the MIME type of the HTTP response with the `setContentType` method:

```
response.setContentType("text/html");
```

To send information to the user's browser, we use `response.getWriter()`, which returns a `PrintWriter`:

```
PrintWriter out = response.getWriter();
```

Assuming that we are sending content of type text/html, we can now use `out.println` to send HTML to the user's browser:

```
out.println("<HTML>");
out.println("<Head><Title>Hello, world</Title></Head>");
```

Apache

We could continue to use Tomcat as our main HTTP server. However, it is neither as fast nor as configurable as Apache or most other servers. For this reason, it's typical to use Apache for most HTTP requests and to forward servlet- or JSP-related requests to Tomcat.

In order for Apache to communicate with Tomcat, we must compile a module into our Apache server. The traditional way was with **mod_jserv**, based on a project called JServ. A new module, known as **mod_jk**, compiles into your Apache server similarly to `mod_jserv` but uses a more efficient and flexible protocol to communicate with Tomcat.

The easiest way to install `mod_jk` is to download the source code for Tomcat from the Jakarta web site. Even if you have downloaded the binary version of Tomcat for general use, you will need to retrieve the source code in order to compile and install `mod_jk`. After you unpack the source distribution, change to `src/native/apache1.3`. If you are using an early version of Apache 2.0, you should go to `src/native/apache2.0` for the `mod_jk` source code.

The following instructions assume you have built your Apache server with the ability to handle DSO, dynamically loaded modules that were not originally compiled into the server. DSO is a wonderfully flexible mechanism for adding new modules; not all modules can always handle this flexibility, and you might find yourself compiling some or all of your Apache server statically. While `mod_jk` can certainly be compiled statically, the on-line documentation encourages users to install it as a DSO, both because it is easier to do so and because it means that you can update `mod_jk` without having to recompile Apache.

Compiling a module as a DSO means it must link against the Apache server using the names and addresses that were specified at the server's time of compilation. In order for us to compile modules with the same environment and information as the server, Apache provides **apxs**, a Perl program that ensures our modules are compiled correctly. `apxs` takes the same arguments as `cc`, as well as several of its own that allow us to install the module automatically.

From inside the `apache1.3` directory, we can compile `mod_jk` in `httpd.conf` with the following command:

```
/usr/local/apache/bin/apxs -i -o mod_jk.so -I../jk \  
-I$JAVA_HOME/include \  
-I$JAVA_HOME/include/linux -c *.c ../jk/*.c
```

If you enter the above command in three lines, rather than one, remember to include the backslashes (\) as the final character on each of the first two lines. Also note that we did not include the -a option, which activates the module in the Apache configuration file, because (as we shall soon see) that is done from within another, automatically generated, configuration file.

Now that mod_jk is installed, we must get Tomcat and mod_jk to speak to each other. Normally, Tomcat expects to receive requests from Apache with the Ajpv12 protocol. However, mod_jk and Tomcat both understand the Ajpv13 protocol, which is more advanced in a number of ways. We will thus need to modify our Tomcat configuration so that it supports Ajpv13, and then configure Apache to use that protocol to communicate with Tomcat.

Tomcat has two configuration files, one for the HTTP server (web.xml) and one for the Java servlet container (server.xml) files. Even if you have never worked with XML, you should not have to worry very much; not only is XML easy to learn, but the Tomcat configuration files are heavily commented. Both configuration files are in the \$TOMCAT_HOME/conf directory.

In order to tell Tomcat to use Ajpv13, we must find the section of server.xml that defines the Ajp12 connector. The section normally looks like this when you install Tomcat:

```
<ConnectorclassName="org.apache.tomcat.service.  
    <Parameter name="handler"  
        value="org.apache.tomcat.service.connector.  
    <Parameter name="port" value="8007"/>  
</Connector>
```

As you can see, this defines the TCP/IP connector handler and indicates that Tomcat should use the Ajp12ConnectionHandler object in the org.apache.tomcat.service.connector package. We will add a similar block immediately following the one shown above:

```
<Connector className="org.apache.tomcat.service.  
    <Parameter name="handler"  
        value="org.apache.tomcat.service.connector.  
    <Parameter name="port" value="8009"/>  
</Connector>
```

Aside from changing the name of the handler object, you can see that we have modified the port number to be 8009.

The `mod_jk` instructions make it clear that we should add the new `Ajp13` handler to `server.xml`, leaving the `Ajp12` handler in place. Otherwise, there may be problems when you try to shut down Tomcat.

Shut down Tomcat with `$TOMCAT_HOME/bin/shutdown.sh`, and start it up again with `$TOMCAT_HOME/bin/startup.sh`, just to make sure that the configuration changes did not break anything. If all is fine, you should see messages indicating that the `HttpConnectionHandler` is running on port 8080, the `Ajp12ConnectionHandler` is running on port 8007 and the `Ajp13ConnectionHandler` is running on port 8009.

The easiest and fastest way to tell Apache how to connect to Tomcat is to use **`mod_jk.conf-auto`**, a file that Tomcat generates each time it is restarted. This file, which is located in `$TOMCAT_HOME/conf`, contains all of the Apache directives necessary to load and use Tomcat. You simply need to include this set of definitions from within your Apache configuration:

```
Include /usr/java/jakarta-tomcat-3.2.1/conf/mod_jk.conf-auto
```

`mod_jk.conf-auto` is not only useful and automatic, it also provides a good sense of how to configure `mod_jk` and how to create sophisticated interactions between Apache and Tomcat. One thing to remember when working with Apache and Tomcat is that Apache must always start up after Tomcat is running so it can connect to the appropriate socket.

A Simple Set of Servlets

To demonstrate how easy it is to write servlets, we will create a simple web application—a blog-creation tool. Blogs, or “web logs”, are increasingly popular web diaries in which the newest entries traditionally appear at the top. The first web log was Dave Winer's *Scripting News* (<http://www.scripting.com/>), but there are many thousands of web logs that provide useful news and commentary on a variety of topics.

We will use servlets to create a very simple web log. The actual log entries will be stored in a PostgreSQL database, which we can define as follows:

```
CREATE TABLE BlogEntries (
  entry_id      SERIAL      NOT NULL PRIMARY KEY,
  entry_date    DATETIME    NOT NULL CHECK
  entry_headline TEXT       NOT NULL CHECK
  entry_text    TEXT        NOT NULL CHECK
  UNIQUE(entry_date, entry_headline)
);
```


Since we're going to be retrieving data by date and headline, we create an index on each of two columns:

```
CREATE INDEX headline_date_index ON BlogEntries
CREATE INDEX entry_headline_index ON BlogEntries (entry_headline);
```

Now that we have created our database table and indices, we will need to create two servlets: one servlet will receive input from an HTML form and use that input to insert a new row into the BlogEntries table. (Presumably, this servlet will only be available to the owner of the site, who is the editor of the web log.) The second servlet will retrieve all web log entries from the last three days, displaying them in the traditional last-in-first-printed order.

The servlet for adding a new web log entry, AddBlogEntry [see Listing 3 at <ftp://ftp.linuxjournal.com/pub/lj/listings/issue84/>], expects to receive two parameters from an HTML form. The first parameter (entry_headline) contains the headline, while the second (entry_text) contains the text associated with it.

The servlet in Listing 3 defines an instance variable con which contains the JDBC database connection. The servlet also defines three methods:

- init, which is before the servlet is first executed. In init, we make an initial connection to the database, keeping the connection around for future use.
- doGet, which prints an error message indicating that only POST requests will be honored by this servlet.
- doPost, which uses the database connection established by init to INSERT a new row into the BlogEntries table.

Modifying a servlet is different from modifying a CGI program in that the servlet container must reload the servlet from disk. Apache and mod_perl do not reload Perl modules by default; so too does Tomcat ignore modified servlets by default. You can change this behavior by setting the "reloadable" attribute to "true"; if you fail to do this, you will need to restart Tomcat each time you modify and recompile a servlet. Of course, there is a performance penalty when servlets are reloadable, which is why the Tomcat documentation suggests keeping them nonreloadable in production systems.

Our doPost method is the real workhorse in this servlet, taking input from the user's HTML form and inserting them into our table in PostgreSQL.

First we make sure that we have received the entry_headline and entry_text parameters from the user and the parameters aren't empty. If one or more is empty, then we create a message that indicates what was missing. Otherwise,

we go ahead and create a PreparedStatement for inserting a new row into the database.

Perl programmers will see many similarities between JDBC and Perl's DBI. JDBC requires that we create a statement based on the database connection:

```
PreparedStatement statement =
    con.prepareStatement(
        "INSERT INTO BlogEntries " +
        " (entry_date, entry_headline, entry_text) " +
        " VALUES " +
        " (NOW(), ?, ?)"
    );
```

Since we are using a PreparedStatement rather than a simple statement, we can use question marks (?) instead of variable values. The drivers for some databases, such as Oracle, take advantage of these placeholders and use them for greater speed. But even users of low-end databases can benefit from using placeholders because they ensure strings will be quoted correctly, even if they contain quotation marks or apostrophes:

```
statement.setString(1, entry_headline);
statement.setString(2, entry_text);
```

Notice how the first placeholder is numbered 1, rather than 0. Keep in mind that these two values are strings; if they were integers or floating-point numbers, we would have to use a different method on statement.

Next, we perform the actual insert:

```
int updateCount = statement.executeUpdate();
```

updateCount is assigned the number of rows that were affected by the executeUpdate() method. In this particular case, we were trying to insert a single row, so we compare updateCount with 1. If we were to use executeUpdate() to perform an SQL "UPDATE", updateCount might contain a different number.

Finally, we catch exceptions that might have occurred during our use of SQL. We then print an error message, including the text of the exception. While printing such explicit messages to the end user might not be a good idea on a production web site, it is an excellent idea during development.

Displaying the Web Log

Now that we have seen how a servlet can be used to enter information into our web log, we will write another servlet to display the latest contents. This servlet will be relatively simple; it will take no parameters and will display the latest

contents of the web log (see Listing 4 at <ftp://ftp.linuxjournal.com/pub/lj/listings/issue84/>).

Our ShowBlog servlet will only have two methods, **init** (which is identical to the "init" method from AddBlogEntry) and **doGet**. doGet will retrieve all of the entries in a web log, from the newest to the oldest. It displays each entry as a three-column row in an HTML table, showing the date and time at which it was added, the headline and the text associated with that headline.

Of course, a real web log will do things in a slightly more intelligent way, limiting the number of remarks and arranging them with a better sense of design. But that's easy enough to do once we have retrieved the information from the database in the correct order.

We create our query (inside of a "synchronized" block) and wrap it into a Statement. Notice how we need not use a PreparedStatement because we are not planning to instantiate any variable values into the statement.

We retrieve the results from the query into a ResultSet:

```
ResultSet rs = statement.executeQuery(query);
```

A ResultSet allows us to pull results out of the database one row at a time. We can iterate through each row inside of a while loop using the rs.next() method. Within each iteration, we can retrieve a column as a String value using the rs.getString() method, passing the name of the column as a parameter.

After compiling this servlet and placing it on my system, I was able to add some new web log entries and display them within a matter of minutes.

Conclusion

Servlets are the Java world's equivalent to the Perl world's modules for mod_perl. In many ways, they are actually better as they provide a great deal of power without endangering the web server with potentially risky programs. This month, we saw some simple ways to build web applications using servlets and open-source tools that we can download from the Web. Next month, we will continue our exploration of server-side Java by looking at some simple uses for Java Server Pages, also known as JSPs.

Resources



Reuven M. Lerner owns and manages a small consulting firm specializing in web and internet technologies. As you read this, he should be (finally!) finishing *Core Perl*, to be published by Prentice-Hall later this year. You can reach him at reuven@lerner.co.il, or at the ATF home page, <http://www.lerner.co.il/atf/>.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

Managing Multiple Cooks

Marcel Gagné

Issue #84, April 2001

Marcel offers an intranet recipe called Grand Salmar Station.

Vien ici, François! Have a look at this. What do you mean, what is it? It is our new intranet. What do you think of it? *Mon Dieu!* You cannot tell me because you do not know what an intranet is? I am certainly glad our guests have not yet arrived, François. You know, working here at Chez Marcel, they naturally assume that you are an expert about these things.

An intranet, François, can be thought of as your own private Internet, a networked environment where users can share information, participate in discussions and use networking technologies to make getting to that information as easy as possible. It can be many things, really—a database providing access to corporate documents, an information center for job postings, a discussion board or a place to find the results of the latest hockey pool, *non?* An intranet is a place to share information. It can take many forms, but the essence of an intranet is a high-tech bulletin board, one that lets you post simple notices as well as entire multipage documents. Unlike that corkboard in most company lunchrooms, a good web-based intranet has virtually unlimited room.

François, why are you not looking at me when I talk to you? Ah! *Mes amis.* Welcome to my restaurant. François! To the cellar *pour du vin.* *Vite!* Bring back the 1990 Vosne-Romanée Les Beaux Monts. Nothing like a good Burgundy to discuss networks, *non?* *Merci, François.*

Please sit, *mes amis.* Before you arrived, I was showing François a special feature from this very restaurant. Although we love to bring you delicacies from around the world, sometimes it is our kitchens that do the creating, *non?* Sally Tomasevic, master chef, has written an intranet package we call Grand Salmar Station. What is really wonderful about this package is that it is virtually self-

administering. A common problem with intranet solutions is that they require a technical person to oversee the project. At the very least, someone has to write HTML, maintain the structure and deal with dated information. Unfortunately, sometimes that person (and their dedication) can be hard to come by.

Ah, François, you have returned. *Merci*. Please pour for our guests. What if I told you, *mes amis*, that you could deploy an intranet and turn it over to your users and let them maintain it without having to train them? They would not need to know any HTML, and they would not have to code a single line. Chef Sally has created just such a package. It's even better. Grand Salmar Station will automatically create and maintain all links for you and will even expire old postings or dated information without you lifting a finger. Any newly added item will magically appear on the intranet's What's New page to highlight it. Normally, such adding and deleting requires user intervention and links must be verified and re-created. No problem with this intranet. It does it all for you.

It is a bulletin board, a flexible news center, an internet reference list, an office directory and a document management system, all in one. Best of all, Grand Salmar Station is freely distributed under the GPL.

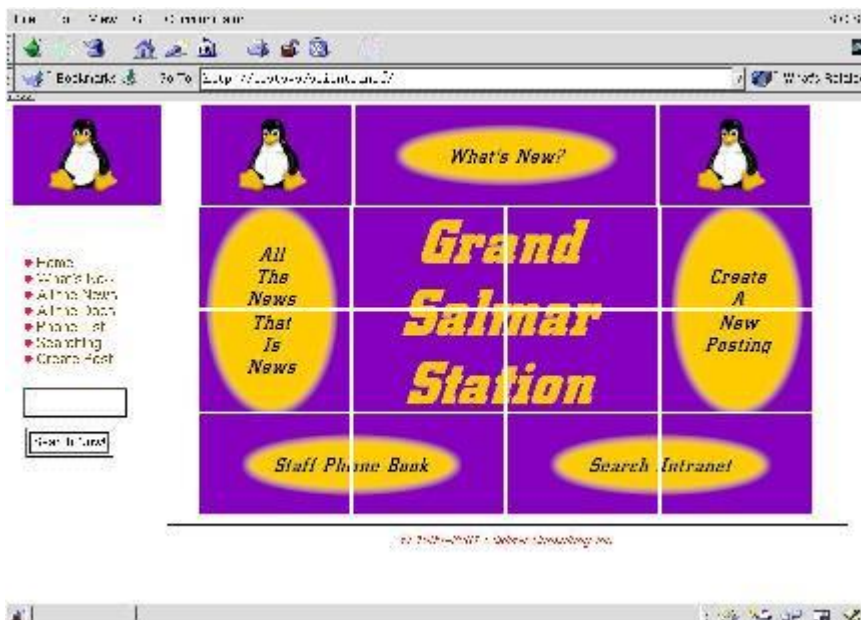


Figure 1. Grand Salmar Station Front Page

For this recipe, you will need the following ingredients: a Linux System (but that goes without saying, *non?*), an Apache web server, Perl, PostgreSQL and the latest Grand Salmar Station source.

La Préparation

Grand Salmar Station is written completely in Perl and uses PostgreSQL as its database. You may recall PostgreSQL from past visits to the restaurant. Our

menu has featured applications based on this excellent database, so you may already be quite familiar with it. It is possible that you even have it running as part of your day-to-day processes. In case you do not, I will give you a quick introduction. For the others, may I recommend a little brie while you wait? You will find cooking instructions for the intranet a little further on.

PostgreSQL is an advanced multiuser, relational database management system (RDBMS) distributed freely along with its source code. Originally written in 1985 and worked on by many developers worldwide, PostgreSQL is fast, powerful, supports most (if not all) SQL standards, and best of all, it is free. You probably don't even have to go looking for PostgreSQL since it is usually packaged as part of most major Linux distributions. Look on your distribution CD it is probably already there. In fact, on some systems, PostgreSQL is part of the default install making it that much easier.

Here at Chez Marcel, we enjoy cooking with open source, *non?* And we love working from source. So, for this recipe, we will be building PostgreSQL from the freshest of ingredients. The latest version is available by visiting the site at <http://www.postgresql.org/>, or as I mentioned mere moments ago, simply take it from your own distribution CD.

After downloading the latest bundle, I extracted it to a temporary location, changed directory to the source directory and compiled. Here are the steps:

```
tar -xzvf postgresql-7.0.3.tar.gz
cd postgresql-7.0.3/src
./configure
make
make install
```

The distribution directory (postgresql-7.0.3 in this case) has a nice INSTALL file that you might want to take a moment to read since there are some options related to the configure script that you might find useful. For instance, by default, PostgreSQL will install in the /usr/local/pgsql directory, and you may find that location less than palatable. Tastes vary.

When the compile is done, PostgreSQL will have to know how to find its libraries. You can always modify your environment variable to include /usr/local/pgsql/lib in the LD_LIBRARY_PATH, but it's probably easier to add the path to the /etc/ld.so.conf file. This is a text file that tells the system where to search for libraries. Because it is straight text, just add the path to your libraries and run this command as root:

```
ldconfig
```

If you decided to install a PostgreSQL binary from your CD, a postgres user will have been created as part of the installation. Otherwise, create a postgres user

with its home directory being the PostgreSQL install directory. Then, assign a password to the user and log in as postgres. If you built your database along with me, you will be in the `/usr/local/pgsql` directory. Next step is to create a data directory:

```
mkdir data
```

To initialize the database for the first time, use the following command:

```
$ bin/initdb -D /usr/local/pgsql/data
```

That is, of course, the data directory that you just finished creating, *n'est-ce pas?*

You will see a number of messages going by as PostgreSQL reports on what it is doing. Not quite enough time for a full meal, but perhaps a single escargot and a sip from your wineglass, *non?* Meanwhile, some default permissions will be set. In addition, a default database will be created along with PostgreSQL's own database (`pg_database`) for user and other database information. Several views are then created after which you should get a message like this:

```
Success. You can now start the database server using:  
/usr/bin/postmaster -D /var/lib/pgsql  
or  
/usr/bin/pg_ctl -D /var/lib/pgsql start
```

Either will work, but the second is a better choice because it launches the process into the background for you. You will also want to add this to your startup for system boot.

Now, it is time to create some users that will have access to your database. If you are installing as root (for access to Perl directories, CGI directories, etc.), then root will have to be added, as will the user "nobody". This is often the user that your httpd server runs as. Some have a user called "www" to run web services. I will use "nobody" as mine, and you may use whatever your server is configured for. Start by logging in as your postgres user, and execute the following commands to add users "root" and "nobody" to your PostgreSQL system:

```
$ createuser root
```

You'll be prompted for root's UID (accept the default) and whether user root is allowed to create databases. Answer "y". When asked whether root is allowed to create users, I answered "y" again. Now, do the same thing for user nobody. The only difference is that I answer "n" to the question of whether nobody is allowed to create databases as well. Depending on which version of PostgreSQL you are using, the question of whether a user is allowed to create other users may be worded this way:

Is user `whoever` a superuser?

The answer is still “n” or “no”.

And Now, Your Perl Modules

Now, onto the Perl side of things. In order to make programming database access easier with Perl, the DBI modules were developed. DBI stands for Database Interface and consists of a collection of routines that offer standard hooks into an SQL database. DBD is an application program interface (API) for Perl 5 to interface with database systems. The idea is to provide a consistent set of modules and calls so that database access code is portable without too much fuss.

Now, since there are a number of database formats out there (PostgreSQL, MySQL, Sybase, Oracle, etc.), there will be some variance in accessing and talking to those various databases. This is where the DBD modules come into play. DBD are the Database Dependent modules. They are identified by the database they support by a simple suffix. For instance, DBD-Pg is the DBD module for PostgreSQL. Meanwhile, DBD-Informix is the DBD module for the Informix database.

Essentially, the DBI module is common to all the various databases, but the DBD module must be (and is) database-specific.

Downloading and Installing the Modules

Both the DBD and DBI module can be found at the CPAN FTP site, a huge Perl resource on the Web. Because the two modules are in slightly different directories, I will give you both starting with the DBI module. See <ftp://ftp.cpan.org/CPAN/modules/by-module/DBI/>.

The DBD module can be found at the CPAN site at <ftp://ftp.cpan.org/CPAN/modules/by-module/DBD/>. At the time of this writing, the latest and greatest DBI version was DBI-1.14, whereas the DBD release number was DBD-Pg-0.95. It probably makes sense to get the latest and greatest when it comes to these modules, particularly if you are using the latest PostgreSQL. The reason for this is that the modules grow and develop with the databases they reference. The most recent PostgreSQL database is best served by the most recent DBD-Pg.

Install the DBI module first by unpacking the distributions into some temporary directory and following these steps:

```
cd /usr/local/temp_dir
tar -xvzf DBI-1.14.tar.gz
cd DBI-1.14
perl Makefile.PL
```

```
make
make test
make install
```

To install the DBD module, the process is similar. Please note that more recent versions of the DBD module now require you to set a couple of environment variables before the install can occur. These are `POSTGRES_LIB` and `POSTGRES_INCLUDE`:

```
export POSTGRES_LIB=/usr/local/pgsql/lib
export POSTGRES_INCLUDE=/usr/local/pgsql/include
```

Now, you can run the install:

```
cd /usr/local/temp_dir
tar -xzvf DBD-Pg-0.95.tar.gz
cd DBD-Pg-0.95
perl Makefile.PL
make
make test
make install
```

Installing the Intranet Software

Ah, *la pièce de résistance*. Start by downloading the latest release from the Salmar web site at <http://www.salmar.com/gss/>. Then, extract it into a temporary directory. Once again, it comes with a lovely little README file that you may want to indulge in. For the impatient, this is all you do:

```
tar -xzvf sciiutra.tar.gz
cd installintra
```

Before we begin the actual install, we must pause and modify one little configuration script. This is a small text file called `sciiutra_conf.pl`, and you can edit it with your favorite text editor. Here is what you will find in the file.

Listing 1. Contents of `sciiutra_conf.pl`

The first thing to do is make sure that the path to your Perl executable is correct. It will either be `/usr/bin/perl` or `/usr/local/bin/perl`. Now, go down to the last three lines and fill in the path to your Apache server's document root and `cgi-bin` directory. What you see above is the file's default configuration. If this fits your own system configuration, then you don't have to change anything.

Now, it is time to finish our install:

```
./install
```

Right away, the script does a couple of sanity checks, specifically to make sure that you have PostgreSQL running and Perl installed. Once satisfied, it will ask you an interesting little question:

```
Do you also wish to install the phonebook (Y/N) ?
```

A call for you, perhaps? Can you hear the music, *mes amis*? Ah, memories. Think back to the June 2000 issue of *Linux Journal*. Chez Marcel's little kitchen turned up an intranet telephone book for the "Who's Who in Linux?" issue. Well, if you have our little phonebook installed, then you can reply with an "N" for no, otherwise, why not install this wonderful little phonebook? It will then be available from the Grand Salmar Station Intranet menu, and it will be wonderful, like this lovely little Syrah. *Pardonnez moi* while I have a little sip. Ah, wonderful.

Next, you will be asked whether you wish to install the intranet. Well, *mes amis*. That is why we are here, *non*? Just respond with a "Y". The install then calls a Perl script to do the rest of the installation. You are almost done, *mes amis*. One more thing.

As I mentioned in the introduction, Grand Salmar Station can automatically clean up after itself. If you decide to post a date-sensitive item, the software can automatically remove this item from the intranet after a specific date. To activate this feature, add the following command to run with **cron**:

```
/usr/local/apache/cgi-bin/scimaint/dailyclean.pl
```

You may, if you wish, copy it to some other location. We will discuss the security aspects of the administrator shortly.

Working with the Intranet

There are two different faces to the intranet. What everyone else sees and has access to and a few little items strictly for the administrator's eyes. The administrator's menu is available via your favorite browser at `http://yourmachine_name/scimaint/admin/`.

From here, you can set various configuration parameters. The first option allows you to create, modify or remove posting categories. You'll see that as "Update Categories" on the menu. The Posting Category Heading is what your users will see when they look at the public face of the intranet. The default category is "News", but you can set it to documents. News items will show up in the user menu under "All the News" while documents will show up under "All the Documents". Both will appear briefly in the "What's New" menu, but you can decide to override that default at this point.

You can also define who is allowed to post here. Your choice is between allowing only administrators or making it a public category and letting everyone have a go. Have a look at Figure 2 for a special category created by your chef.

Figure 2. Creating Posting Categories

Another choice you have at this point is whether to allow internet links. Links can refer to sites anywhere else on the Internet, internally, or for that matter, company documents somewhere on your local network. You can then decide on posting order: whether alphabetically or by date, ascending or descending. Finally, you specify whether the system will take care of expiring and removing these links automatically.

Grand Salmar Station offers many other options. For instance, you can change the style of the intranet—in essence, the look and feel of it. You will find six styles included with the distribution with such exotic names as “Beach Time”, “Copper Kettle” and “Blue Skies”. *Mais oui*, of course you can create your own. Any style you choose can then be propagated across your entire intranet.

The phonebook can also be administered from here (if you chose to install it) as can other intranet defaults like expiration dates. Explore. Enjoy. And, *bien sur*, be sure to read the accompanying documentation, also available from the Administrator's menu.

If you want to get a feel for what a user will see, use http://yourmachine_name/scimaint/admin/demoreset.html to generate a demonstration database along with some sample entries. *Ah, oui*. The users—they have their own path into the intranet: http://yourmachine_name/sciintranet/.

Have a look at Figure 3 for a peek at their “What's New” screen.

Figure 3. What's New on the Intranet?

Mon Dieu, mes amis. It is late, *non?* François, please, refill our guests' wineglasses before they go! I hope, *mes amis*, that you will enjoy working with our intranet. Sally and I are quite proud of it, and we hope you will find it useful as well. In the meantime, I must thank you for coming tonight. Finish your wines, relax and make sure you join us next time at *Chez Marcel*. Your table will always be waiting.

A votre santé! Bon appétit!

Resources



Marcel Gagné lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting, Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy and is coeditor of *TransVersions*, a science fiction, fantasy and horror anthology. He loves Linux and all flavors of UNIX and will even admit it in public. In fact, he is currently working on *Linux System Administration: A User's Guide*, coming soon from Addison-Wesley Longman. He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things from his web site at <http://www.salmar.com/marcel/>.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Battening down the Hatches with Bastille

Mick Bauer

Issue #84, April 2001

How to harden your system.

Are you one of the many people who, when running `ps -ef` for the first time after installing Linux, has no idea what half of those processes are? Don't be embarrassed; we all have to start somewhere, and it takes time (and a lot of man-page lookups) to understand the myriad of applications and daemons it takes to make a UNIX system run. While there's no substitute for doing those man-page lookups and truly understanding our systems, we shouldn't punish ourselves by running insecure systems until we've achieved guru status.

Bastille Linux, a powerful set of system-hardening Perl scripts, secures Linux systems and educates their administrators: when run in interactive mode it asks clear, specific questions that allow it to create a custom security configuration for your system. It also explains each question in detail so that by the time you've finished a Bastille session, you've learned quite a bit about Linux/UNIX security.

If you already understand system security and are only interested in using Bastille to save time, you can run Bastille in its "explain less" mode that asks all the same questions but skips the explanations. If you're running Bastille on a "clean" Linux installation, you can even skip the questions altogether and choose one of several "secure by default" security templates depending on what type of system you want. And if you're a real cowboy/girl, you can write your own Bastille configuration template from scratch (or, more likely, by tweaking one of the provided templates to fit your needs).

Do We Really Need Bastille? Yes!

You may be thinking "Why does so much stuff have to be running and enabled by default? Isn't it silly to use special scripts just to trim the fat, when we could simply leave out all that fat in the first place?"

In my opinion most distributions of Linux do have too many things enabled by default. But the fact is that ever-increasing numbers of Linux users are absolute beginners, and if the first time they install and reboot their Linux system, it doesn't do very much (or worse still, doesn't work at all), then fewer people will stick with Linux and learn to run it securely.

In other words, Linux packagers (the people who create distributions such as Red Hat, Caldera, etc.) usually prefer to err on the side of usability rather than on security. Personally I wish more distributions offered both maximum functionality and "paranoid" configuration options in their installers. (Note that Red Hat 7.0's installer does in fact offer such options, although their "paranoid" option isn't necessarily as secure as a post-Bastille configuration.)

How Bastille Came to Be

The original goal of the Bastille team (led by Jon Lasser and Jay Beale) was to create a new secure Linux distribution based on Red Hat. The quickest way to get their project off the ground seemed to be to start with a normal Red Hat installation and then to "bastillify" it (my term) with Perl scripts.

Before long, the team had decided that a set of hardening scripts that could be used on different distributions would be less redundant and more flexible than an entirely new distribution. Rather than moving away from the script approach altogether, the Bastille team has instead evolved the scripts themselves.

The Perl scripts that comprise Bastille Linux are quite intelligent and make fewer assumptions about your system than they did in the days when Bastille was meant only to be used on fresh installations of Red Hat. Your system need not necessarily be either a "clean install" or even Red Hat for Bastille to work, thanks to new functionality in version 1.1.x that transparently gleans a good deal of information about your system before making changes to it.

Obtaining and Installing Bastille

Okay, you're psyched and ready for automated, educational system-hardening. One little warning before we proceed: your mileage may vary. Although Bastille can be used on most Linux distributions, it started out as strictly a Red Hat utility and is still optimized for Red Hat and Red Hat-derived distributions. I'll offer some tips on using Bastille on non-Red Hat-derived distributions, but I can't guarantee your results. When in doubt, refer to Bastille's web page (see Resources).

Speaking of which, that's the home and definitive source of both Bastille Linux and its documentation. A link to the latest version of Bastille Linux can always be found in big bold letters near the top of the home page at <http://>

www.bastille-linux.org/. Once you've downloaded the tarball, move it to /root and unpack it:

```
tar -xzvf ./Bastille-X.Y.Z.tgz
```

That's it, it's installed!

Note that Bastille expects to live in /root. You can probably hack the Bastille scripts to reflect some other home, but I don't recommend it as this is an unsupported action (and affects a lot of scripts). This shouldn't bother you; we usually don't get to choose where RPMs and other software packages get installed either.

Also, you need to have the Perl 5 scripting language installed in order to run Bastille Linux. To check your system for Perl (and its version) simply enter the command **perl -v**. If Perl replies with a version number less than 5.0, or if your system replies with **perl: command not found** then you need to upgrade or install Perl. No current Linux distribution lacks a Perl 5 package; refer to your Linux CD-ROMs or your distribution's home page to obtain a binary package for your system.

Getting Started with Bastille

Using Bastille is easy. From /root/Bastille, you run the script **InteractiveBastille.pl** (see Figure 1), which asks you a long list of questions about what you need enabled and how you need it configured for a balance of functionality and security appropriate for your particular needs. These questions are split into different sections: IPChains, PatchDownload, FilePermissions, etc. The results of this Q&A session are stored in a file called config.



Figure 1. InteractiveBastille.pl

Next, run the script **BackEnd.pl**, which invokes the scripts corresponding to each section in **InteractiveBastille.pl**, using **config** to define the numerous variables that determine how these scripts behave. Depending on how you answered Bastille's questions, some of the component scripts may not be invoked at all (e.g., if you answered "no" to "Configure IPChains?").

If you don't want to bother with all those questions, you can instead run **AutomatedBastille.pl**, which will give you the choice of several default security baselines and will then immediately harden your system accordingly. **AutomatedBastille.pl** is an extremely simple script; really, it's just a mechanism for invoking **BackEnd.pl** with a canned configuration file.

The included baselines (**Default_Workstation** and **Default_Workstation_plus_Firewall** in Bastille v.1.1.0) can easily be adapted for your own particular needs; thus, it's a simple matter to create your own baselines and add them to **AutomatedBastille.pl** if you have large numbers of systems to harden. Alternatively, you can skip **AutomatedBastille.pl** altogether and simply run **BackEnd.pl** with the configuration file of your choice, for example:

```
./Backend.pl ./myconfig /root/bastille-output-log
```

Some Notes on InteractiveBastille.pl

InteractiveBastille.pl explains itself extremely well during the course of a Bastille session. If you take the time to read this script's explanations of its own questions, you'll learn a lot about system-hardening. If you already know a lot, you can select the explain less option at any point to make the questions a bit less wordy (and if you change your mind, you can choose "explain more" later).

Bastille's verbosity notwithstanding, the following general observations on certain sections may prove useful to the beginner:

- Module 1: IPChains.pm—IPChains is Linux's firewall system. If your host is going to be accessible by hosts on the Internet, I strongly recommend that you configure IPChains. Even a few simple packet-filtering rules will greatly enhance overall system security.
- Module 2: PatchDownload.pm—if you have a Red Hat system, Bastille can download and install RPMs of any software that has changed since you originally installed it.
- Module 3: FilePermissions.pm—this module restricts access to certain utilities and files, mainly by disabling their SUID status. Generally speaking, SUID is used to allow a process to behave as though it had been invoked by root, allowing nonroot users to use utilities such as **mount**, **ping** and **traceroute**. However, these utilities are usually not needed by unprivileged users and can in fact be used for all sorts of mischief. Disabling SUID status makes such commands usable only by root.
- Module 4: AccountSecurity.pm—this module allows you to create a new administration account and generally tighten up the security of user-account management. These are all excellent steps to take; I recommend using them all.
- Module 5: BootSecurity.pm—if it's possible for unknown or untrusted persons to sit in front of your system, reboot or power-cycle it and interrupt the boot process; these settings can make it harder for them to compromise the system.
- Module 6: Securelnetd.pm—in this section, internet services are tightened down and warning banners created.
- Module 7: DisableUserTools.pm—disabling the compiler is a good idea if the system's nonroot users don't explicitly need to use it. As in most other cases, when Bastille says "disable" here it actually means "restrict to root-access only".
- Module 8: ConfigureMiscPAM.pm—several useful restrictions on user accounts are set here.
- Module 9: Logging.pm—too little logging is enabled by default on most systems. This module increases logging and allows you to send log data to

a remote host. Process accounting (i.e., tracking all processes) can also be enabled here but is overkill for most systems.

- Module 10: MiscellaneousDaemons.pm—in this section you can disable a number of services that tend to be enabled by default despite being unnecessary for most users.
- Module 11: Sendmail.pm—self-explanatory.
- Module 12: RemoteAccess.pm—if you don't have it yet, Bastille can download and install the Secure Shell for you! SSH is a secure replacement for **Telnet**, **rsh** and **rlogin**. Note that Bastille will attempt to install RPMs compiled for Red Hat systems on i386 architectures. If you run Linux on a non-PC-compatible architecture or use a distribution that chokes on Red Hat RPMs (e.g., Debian), then this module won't work for you.

Logs and Running Bastille on Non-Red Hat Distributions

So, after InteractiveBastille.pl creates config and BackEnd.pl implements it, what then? How do I know what happened? Thanks to Bastille's excellent logging, it's easy to determine exactly which changes were successful and, equally important, which failed, and that makes this a good place to discuss running Bastille on non-Red Hat distributions.

As I mentioned earlier, while Bastille is optimized for Red Hat and Red Hat-derived Linux distributions (e.g., Mandrake, Yellow Dog, etc.), it is intelligent enough to work on others as well. And in fact, my own experience bears this out: I ran Bastille 1.1.0 on a system running SuSE Linux 6.3, and I'm pleased to report that far more of Bastille worked than not.

The first thing I noticed during my SuSE-6.3 Bastille session was that several actions are explicitly Red Hat-centric (InteractiveBastille.pl gives warning to that effect). Module 2, PatchDownload.pm, is strictly for Red Hat systems. And as noted above, Module 12, RemoteAccess.pm, installs the Red Hat i386 version of Secure Shell 1.2.27.

One other question in my session, regarding whether I should run **named** in a chroot jail [see "Paranoid Penguin: Securing DNS and Bind", *LJ* October 2000] came with the same warning. Luckily, both downloading patches and configuring named to run chrooted are things one can do manually without too much trouble.

Although these were the only explicitly Red Hat parts that Bastille warned me about during the InteractiveBastille.pl session, there were other parts of Bastille that didn't deal with SuSE too well. BackEnd.pl returned no errors (to the console) at all; I wasn't aware that anything had failed until I read Bastille's logs.

Note that it's probably a good idea to review these logs regardless of whether you think something's gone wrong; meaningful logging is one of Bastille's better features. And whether a beginner or a security guru, you should know not only what changes Bastille makes, but how it makes them.

Logically enough, Bastille writes its logs into `/root/Bastille/log/`. Two logs are created by `BackEnd.pl`: `action-log` and `error-log`. `action-log` provides a comprehensive and detailed accounting of all of Bastille's activities. Errors and other unexpected events are logged to `error-log`.

Most of the errors in my SuSE Bastille session involved files not residing where Bastille expected them. In a few cases this was due to SuSE keeping them in different places than Red Hat does; for example, Red Hat puts Apache (web server) configuration files in `/etc/httpd/conf`, whereas SuSe puts them in `/etc/httpd`. Bastille therefore failed to make any changes to Apache on my SuSE system.

In this example and most other file-location errors, it was fairly easy to figure out how to make the changes manually; Bastille's list of questions, `/root/Bastille/Questions.txt`, provides plenty of hints (easily found via **grep**), and if you understand Perl you can parse the scripts in `/root/Bastille/Bastille` to determine precisely what Bastille was trying to do.

The easiest thing to do, however, is simply to change the paths in the Bastille scripts and then rerun `BackEnd.pl`. Many of the paths that I needed to change for SuSE were in `/root/Bastille/Bastille/FilePermissions.pl`. For each file identified by `error-log` as missing, I simply checked for it using **which**. If it did indeed exist but in an unexpected location, I edited `/root/Bastille/Bastille/FilePermissions.pl` accordingly.

In the example below (an excerpt of the real `error-log` on my SuSE system) we see that Bastille couldn't find, `setserial`, `chkconfig` or `ifdown`:

```
#ERROR: chmod: File /bin/setserial doesn't exist!  
#ERROR: chmod: File /sbin/chkconfig doesn't exist!  
#ERROR: chmod: File /sbin/ifdown doesn't exist!
```

I happen to know that `chkconfig` and `ifdown` don't exist on SuSE systems; they're unique to Red Hat and its derivatives. Therefore those particular "errors" were moot. What about `setserial`? Entering the command:

```
which setserial
```

returned the output:

```
/sbin/setserial
```

Then, entering the command:

```
grep setserial /root/Bastille/Bastille/*
```

returned the output:

```
/root/Bastille/Bastille/FilePermissions.pm: &B_chmod(0750,"/bin/setserial");
```

The which command told me where setserial lives on my system; the grep command told me which Bastille module and even which line I had to edit to fix setserial's path.

For a relatively small number of nonexistent files this was no big deal; for each file I determined whether it existed on my system and if so, where. I then verified that FilePermissions.pm was the offending module and edited it accordingly. Doing this for all the files cited in error-log and then rerunning BackEnd.pl took less than 20 minutes.

There was one other problem that needed fixing, and it's illustrated in the following example:

```
#open /etc/httpd/conf/httpd.conf.bastille failed...
#open /etc/httpd/conf/httpd.conf failed.
Couldn't replace line(s) in /etc/httpd/conf/httpd.conf
#open /etc/httpd/conf/httpd.conf.bastille failed...
#open /etc/httpd/conf/httpd.conf failed.
Couldn't replace line(s) in /etc/httpd/conf/httpd.conf because open failed.
```

As you can see, this was caused by the aforementioned difference between Red Hat's and SuSE's Apache environments. Again I used my handy grep command:

```
grep httpd.conf /root/Bastille/Bastille/*.pm
```

This returned several lines:

```
API.pm: $GLOBAL_FILE{"httpd.conf"}=
API.pm: $GLOBAL_FILE{"httpd_access.conf"}=
API.orig.pm: $GLOBAL_FILE{"httpd_access.conf"}=
Apache.pm: my $httpd_file=$GLOBAL_FILE{"httpd.conf"};
```

This told me that Bastille's Apache module references a variable called GLOBAL_FILE to store httpd.conf's path, and this variable is set in API.pm. Changing this path in API.pm and again rerunning BackEnd.pl (with everything commented out except the line invoking Apache.pm) was a trivial matter.

This may seem like a lot of tinkering for the sake of paranoia. But, even if Bastille succeeds in completing only 80% of its tasks on a non-Red Hat system, that system is still significantly more secure than it was before, right? Absolutely.

But Murphy's Law tells us that one of those utilities whose permissions Bastille couldn't tighten down could very well be the one that a mischievous user exploits to grant himself or herself root access. Taking the time to identify and correct these Bastille “hiccougs” pays off, especially if you've got more than one system to harden. Obviously, the more systems of a given type that need hardening, the bigger the return on your time investment in customizing Bastille for that type.

Hooray, I'm Completely Secure Now! Or Am I?

Okay, we've carefully read and answered the questions in `InteractiveBastille.pl`; we've run `BackEnd.pl`; we've checked Bastille's work by going over its logs, and we've gone back and rerun `BackEnd.pl` after tweaking a module or two. Are we there yet?

No, nor will we ever be! Security is a process, not a product. The surest way to create a vulnerable system is to plug it in, turn it on and forget about it, even if at some point prior to forgetting about it you hardened it.

Oops, I'm preaching—force of habit (I do this stuff for a living). Well, even in the immediate task of hardening our system, after running Bastille we've still got a few tasks left before taking a breather.

Remember back at the beginning of the article when I started preaching about the big trade-off that comes with functionality? One of the many ramifications of this is that even Bastille can only do so much; it's impossible to anticipate and create Bastille modules for every software package that might be installed in any given Linux distribution.

Some distributions, Debian and SuSE in particular, include an enormous number of packages—so many that in a recent Slashdot interview Bastille's Jon Lasser (see Resources) referred to them as being especially problematic as a result, from a security standpoint. He hastened to point out that this doesn't mean that systems with a lot of different software packages can't be secured; it simply means that it takes more work.

Therefore, be sure to do the following after running Bastille:

Disable Any Remaining Unnecessary Services: Check which services still have startup scripts in `/etc/rc.d/rcX.d`, where X is your default runlevel. (You can find your default runlevel with the command `grep initdefault /etc/inittab`.)

Startup scripts begin with a capital S. If you don't know what “Ssomed daemon” does, try `man somedaemon`. If you don't need it, then `mv Ssomed daemon dis_Ssomed daemon` (if it doesn't begin with S, the script won't be run at startup).

Take the Time to Learn and Secure Your Applications: Bastille makes it possible to secure a system before knowing very much about system security—that's why its creators have spent so much time adding educational content to it. But both the applications it secures and especially the ones it doesn't won't remain secure unless you invest the time needed to understand them.

BIND, for example, which provides DNS (name) resolution to network applications, has a variety of security features. Bastille configures some of them (on Red Hat and Mandrake, that is) but only scratches the surface. The *Bind Operators' Guide* and the `named` man page are required reading for any system administration who runs BIND, period. Put another way: RTFM!

Disable Any Remaining Unused User Accounts: One of SuSE's more annoying quirks is the inclusion of a long list of entries in `/etc/passwd` for application-specific user accounts regardless of whether those applications are even installed. While very few of these are privileged accounts, many can be used for interactive login (i.e., they specify a shell rather than `/bin/false`).

This is hardly unique to SuSE. Therefore, be sure to check `/etc/passwd` and comment out any unnecessary entries. If in doubt, change the final field (default shell) from a real shell to `/bin/false`—only actual user accounts need shells!

Maintain Your System Software: I've said it at least twice already, but there's no substitute for keeping current. Keep abreast of your distribution's security patches and bulletins! And as you learn more about Linux security, make sure you apply that knowledge to the stuff you installed back when you were a newbie. Do not hook your machine up to the Internet, power it up and forget about it.

Install an IDS: As soon as possible after installing the operating system, install Intrusion Detection software like **tripwire** or **snort**. It's the simplest thing you can do to minimize your chances of finding out from the wrong people (FBI, angry system administrators, etc.) that your system has been hacked and used to attack others. See Bobby S. Wen's article "Open-Source Intrusion Detection Tools for Linux" (*LJ* October 2000).

Monitor Your Logs: The last system-hardening tidbit I'll leave you with this month is the necessity of reading the logs you told Bastille to enhance. These logs are no good to you at all if you never open them.

Parsing log files can be tedious work though, so it's a good idea to either write scripts that periodically check the logs for suspicious activity, or install a tool

like **swatch**, which does for log monitoring what Bastille does for system tightening. In fact, it's worth its own Paranoid Penguin column....

Resources



Mick Bauer (mick@visi.com) is a network security consultant in the Twin Cities area. He's been a Linux devotee since 1995 and an OpenBSD zealot since 1997, taking particular pleasure in getting these cutting-edge operating systems to run on obsolete junk. Mick welcomes questions, comments, and greetings.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

XFree86 and Video4Linux

Robin Rowe

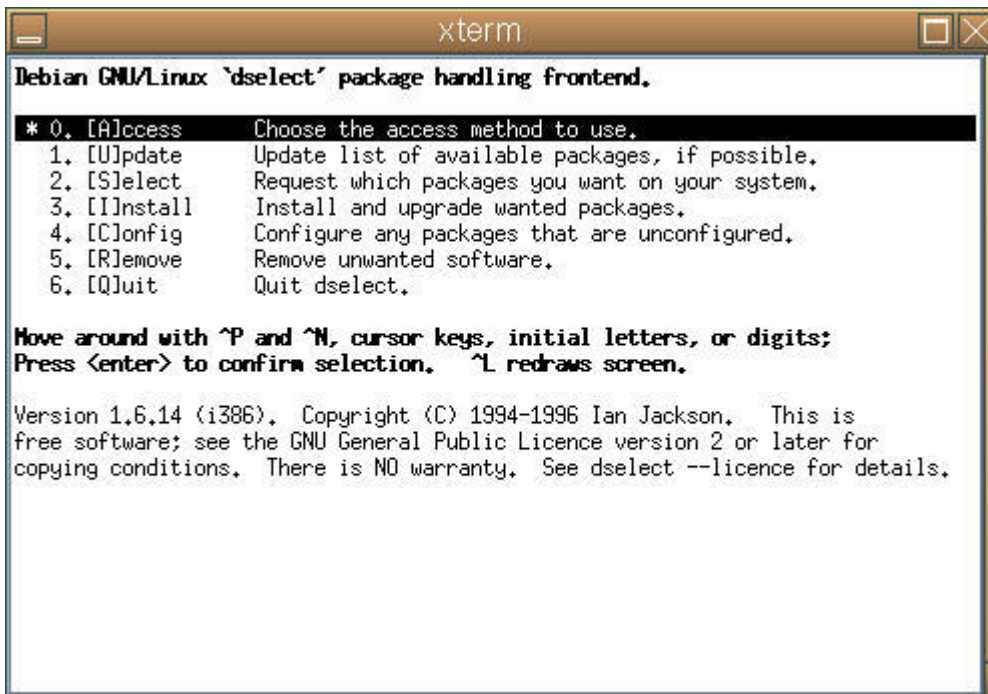
Issue #84, April 2001

Now you can watch cable TV on an X desktop.

Last month we installed Debian Linux 2.2.r2 (aka potato) and configured LILO to boot Linux, Win2K, WinNT, Win98SE and Win98. Building a Linux desktop machine is a two-step process. The second part, installing XFree86 for a graphical desktop, is our first task this month. Once we have the X Windows System working properly we will patch kernel 2.2.17 to install Video4Linux and use XawTV to watch cable television on our X desktop. As a newbie to Linux I will encounter many obstacles doing all this.

By default Linux boots in console mode. Like the old Windows 3.1, a Linux box first boots as a text console then loads the GUI. If you are building a Linux server, the console may be all you need or want. A GUI can create a lot of unnecessary overhead and more can go wrong. However, we are building a desktop. We will install the X Windows System, the Blackbox window manager, Netscape and other graphical applications such as The GIMP.

The Debian package management tool is called **dselect**. This software will install applications included in our six Debian CDs and can also retrieve packages across the Internet. There are about 4,000 packages to choose from on the Debian CDs. Linux applications don't come enclosed in a nice graphical install program the way Windows programs typically do.

The image shows a terminal window titled 'xterm' displaying the 'dselect' package handling frontend. The window has a title bar with standard window controls (minimize, maximize, close). The main content is a text-based menu with the following items:

```
Debian GNU/Linux 'dselect' package handling frontend.
* 0. [A]ccess      Choose the access method to use.
  1. [U]pdate      Update list of available packages, if possible.
  2. [S]elect      Request which packages you want on your system.
  3. [I]nstall     Install and upgrade wanted packages.
  4. [C]onfig      Configure any packages that are unconfigured.
  5. [R]emove      Remove unwanted software.
  6. [Q]uit        Quit dselect.

Move around with ^P and ^N, cursor keys, initial letters, or digits;
Press <Enter> to confirm selection.  ^L redraws screen.

Version 1.6.14 (i386). Copyright (C) 1994-1996 Ian Jackson. This is
free software; see the GNU General Public Licence version 2 or later for
copying conditions. There is NO warranty. See dselect --licence for details.
```

deselect

The console-based dselect interface was slick in its day but is showing obvious age. It isn't very acceptable by modern interface standards, and with so many packages it is difficult to navigate. You may be offered the option to make dselect operate via a web browser. Don't do it! We couldn't figure out how to make this work (we didn't try very hard) but had plenty of trouble turning it off so we could go back to the console interface.

XFree86 (www.xfree86.org) is an open-source windowing system that runs on Linux and other operating systems. It isn't the only X Windows System server available, but it is popular and free. X is different from Windows. The client/server architecture of X enables users to open a windows session to a remote host. We won't notice that running the client and server on the same machine. XFree86 was recently rewritten for much better screen performance, but we will be installing the older version 3.3.6-11 that comes with Debian Potato. We will save installing version 4.x for a later time. It will prove difficult enough installing the supported 3.x version.

Install XFree86 by selecting it in dselect. Installing XFree86 only unpacks the software. It doesn't configure it. There are several programs available for configuring XFree86. One often suggested is **XF86Setup**, a program that configures X from within a graphical X application. The obvious question to ask is: how does this work if you don't have X configured in the first place? The answer is: better than you would expect but not good enough. Life is hard enough without trying to navigate a GUI without a working mouse. The other major configuration program is a dated console application called **xf86config**. If dselect seems primitive then xfx86config must be prehistoric. You can't even

back up if you make a mistake. Still, it does the job. Keep starting over until you get it right.

Our Hercules Terminator 128 2x/I AGP card is listed as one of the many cards you can select in xf86config (the 2x/I is #370 in the database). We were warned by xf86config not to probe clocks or use any Clocks line with this card, otherwise we could seriously break something or fry our monitor. You can let the program probe try to figure out the card by itself. Along the same lines, you must know the correct specs for your monitor. Our ViewSonic E790 monitor supports a vertical frequency of 50-200KHz and a horizontal frequency of 30-95KHz. We found that out easily enough on their web page.

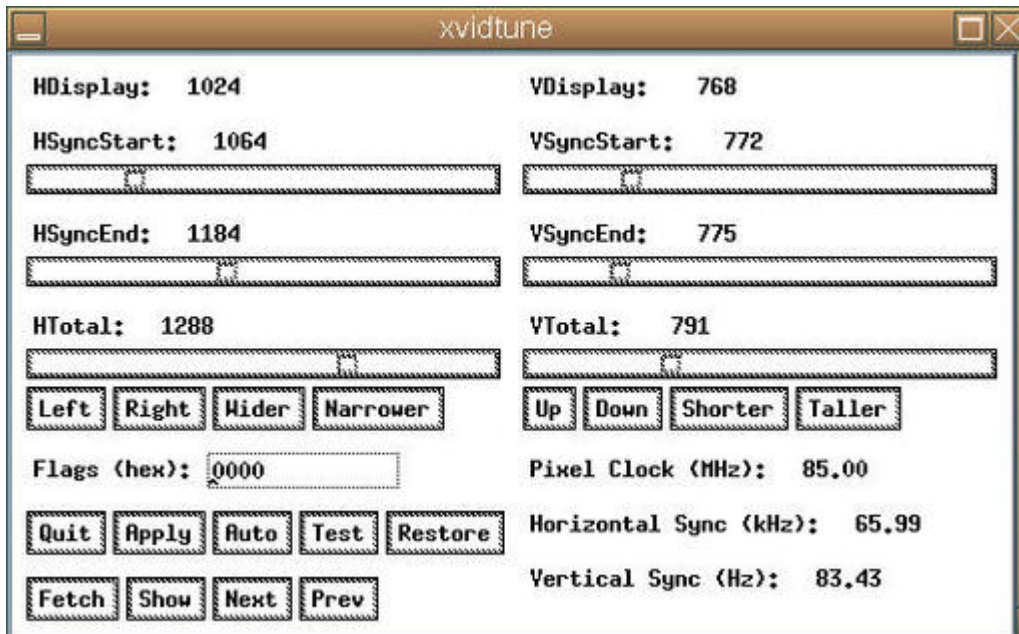
Why our mouse wouldn't work was a mystery at first. Luckily we noticed that gpm, the general purpose mouse or console mouse driver, was one of the packages installed. This will eat mouse events before X can get them. It is possible to configure X to daisy chain with gpm. Not being clear why we want gpm anyway, we simply uninstall it.

Another package we wish we hadn't installed is **xdm**. We installed it without knowing what it was when dselect recommended it with a lot of other X stuff we were installing. What xdm does is automatically launch X when the computer boots, taking you out of console mode immediately and making you log in from X. If you have ever used UNIX in a computer lab you know what this login screen looks like. It is, of course, a disaster if X won't run because you don't have it configured yet! We managed to get into a console prompt and put **exit 0** in /etc/init.d/xdm to stop it on boot.

If you aren't using xdm then you call **startx** to launch X. As X is loading, some console messages flash by before the display flips into graphics mode. If you are having trouble, as we were, you should specify **startx >& x.log** to redirect those messages to a text file you can study later. Pressing Ctrl+Alt+Backspace quits the X server to return to console mode.

Launching X, it comes up in 16-color 640 x 480 mode, definitely not what we asked for. Using Ctrl+Alt+keypad+ is supposed to cycle through the different supported screen resolutions but has no effect. It isn't until we examine our saved x.log file that we realize that we are running the generic X server and not XF86_SVGA as we had specified in xf86config. Nothing warned us that XF86_SVGA wasn't even installed! We check to find that the symbolic link at /etc/X11/Xserver doesn't point to it. Another quick session with dselect sets it right. XF86_SVGA installs separately from the rest of X. We would have caught that if dselect was more user-friendly.

Once we get XFree86 working we need to adjust the frequency settings to center the image properly on our monitor. The **xvidtune** program handles that pretty nicely. You interactively adjust the screen until it matches the appearance you want. However, you can't automatically save that setting as in Windows. You must paste the setting's output by xvidtune into your XF86Config file. Once you get it close using xf86config you will edit the XF86Config file by hand many times making minor adjustments.



xvidtune

A window manager is responsible for the “decoration” or look and feel of window frames in X. Many different window managers are available. Note, we are not talking about desktop managers here, such as GNOME or KDE. They also include a window manager, but desktop managers have a lot of other features such as icons on the desktop, a trash can and a suite of interoperable applications. They also tend to consume a lot of RAM and be slow. We'll give GNOME a try sometime, but for now we're sticking to just a simple window manager.

The default window manager for Debian is Window Maker. It seems okay, but we have a problem with it. Perhaps our problem has a simple solution, but we can't figure it out. How do you take a screen shot? Without that feature it will be pretty hard to write this article!

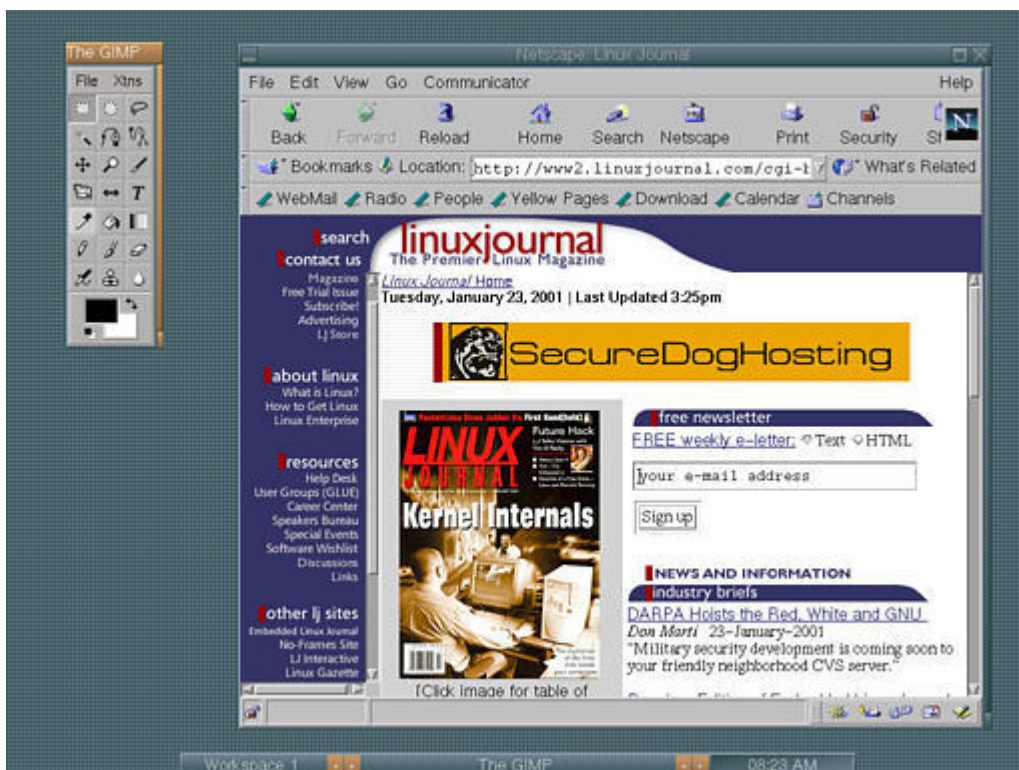
Blackbox (<http://blackbox.alug.org/>) is a window manager known for being lightweight and sexy. It supports themes to allow you to drastically change its look and feel. And, it just seems cool. Using dselect, we install it without difficulty. Window Maker still comes up as our default window manager, but when we right click a menu option has been added to switch to Blackbox.

Window Maker was no slouch, but navigating the same menus in Blackbox is extremely crisp and quick.

To create our screen snapshots we are using The GIMP. We could use **xwpick**, **xgrab**, **xv** or **xdmp**, but The GIMP is an application we wanted to install anyway. It is recognized as one of the best applications available for Linux. The GIMP is very similar to Photoshop, a popular graphics editing program. To take a screen shot choose from the menus "File" "Acquire" "Screen Shot". Doing that works fine in Blackbox, but for some reason fails in Window Maker. The GIMP is open source and available for other operating systems. We use it on WindowsNT, too.

Amaya is an open-source web browser and editor. We've had pretty good luck with it in Windows, but when we install the Linux version, it simply crashes. We're not so interested in Amaya at this point that we feel like tracking down the trouble. Netscape, for license reasons, is not included on the Debian CDs. It must be downloaded from the Netscape site. Although a Solaris version of Internet Explorer is available, Microsoft doesn't offer a Linux version.

We download Netscape for Linux version 4.76 (netscape-smotif-476), the last of the 4.x versions. Netscape 6 is notorious as a resource hog so we avoid it. Version 5 never was. Since we don't have a working web browser in Linux (unless you count the console-mode Lynx browser) we download Netscape by booting Windows and surfing for it using IE. We save the file to our FAT16 Windows partition, then reboot to Linux. In Linux we mount the Windows partition so we can access our download using **mount -t vfat /dev/hdb1 /win98**.



Netscape Screenshot

There is a method to get dselect to install Netscape after you download it, but it seems easier to ignore that and do the standard Netscape install. All nonpackaged installs follow the same general procedure of changing to the generic install directory and uncompressing the tar file using the xvfz options:

```
cd /usr/local/install
tar xvfz filename.gz
```

We then execute the ns-install script to install Netscape. All is not well, however, because we get an error message when we try to launch Netscape. It can't load the libstdc shared library. We are using a different version of that library than Netscape expects to find. We trick it by finding the library we do have and creating a link to give it the same name as the missing (obsolete) library:

```
find / -name "libstdc*" -print
ln libstdc++-3-libc6.1-2-2.10.0.so libstdc++-libc6.1-1.so.2
```

All that remains is to point Netscape to our proxy server using "Preferences" "Proxy" "Manual". We now have a working graphical web browser.

The Hauppauge (<http://www.hauppauge.com/>) WinTV PCI card provides television video in a window. With a price of about \$70 US it has a reputation as one of the best cheap PC TV video cards around. It works in conjunction with your existing video card, but not every card is supported. If you are a channel surfer like me, you may discover you prefer casual TV viewing on your PC. It's convenient, and you can click through the channels much faster. WinTV works with many operating systems including Linux, Windows, FreeBSD and BeOS.



Hauppauge WinTV

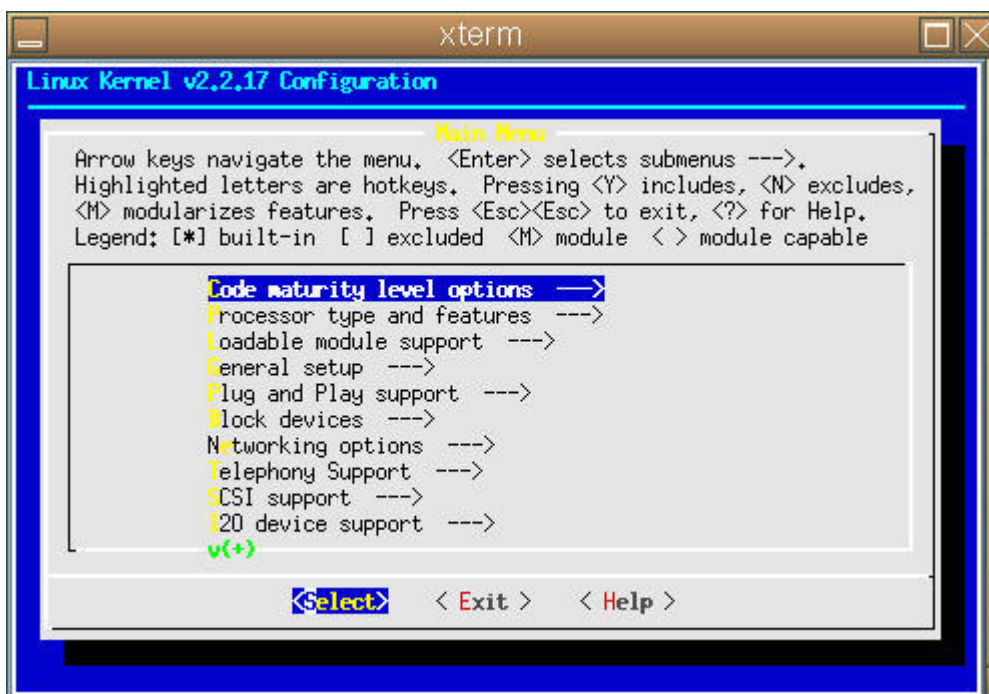
There is a lot of conflicting information on the Web about installation of Video4Linux (V4L) and the bttv driver used with WinTV and other cards based on the BT848 chip. There are so many different driver versions available; that can be confusing. The version of V4L included with Debian Potato is obsolete, so we won't use that.

The current version of V4L requires the i2c bus driver available in the 2.3 kernel. Use **uname -a** to check what kernel version you have. Since our kernel is 2.2.17 we must patch it. We don't understand this right away but know something is wrong because we see a bunch of kcompat.h compile errors attempting to build V4L from source.

Before we can build the kernel we must first install the kernel source using dselect and create a symbolic link:

```
cd usr/src/  
ln -s kernel-source-2.2.17 linux  
cd linux
```

We then do **make mrproper** to delete any old kernel compile settings we might have. We use **make menuconfig** to launch the kernel configuration program. This is a very nice console app. If only xf86config and dselect looked this good! In menuconfig we install kmod (the new module loader) and Video4Linux. Everything we can make as a module (NTFS, VFAT and SMB, but not ext2) we do. There are a lot of options. It takes several tries before we drill through every menu choice and pick the appropriate settings.



menuconfig

Before compiling our kernel we need to install the new i2c driver from source as instructed in the bttv FAQ at www.struse1007.de/bttv/faq.html. The i2c driver supports the internal serial bus protocol used by the BT848 to tune stations and so forth:

```
cd /usr/local/install  
tar xvgz i2c-2.5.4.tar.gz
```

```
cd i2c-2.5.4
vi QUICKSTART
```

To build the i2c driver we replace the i2c* files in the kernel sources with our downloaded driver source then build the driver:

```
make
```

Now it is back to the kernel to build in the new kernel and drivers. We first back up our modules directory. We will build and install all new modules to go with our new kernel:

```
cd /lib/modules
cp -r 2.2.17 2.2.17.bak
cd /usr/src/linux
make dep
make bzImage
make modules
make modules_install
```

Back up the old kernel. Edit lilo.conf to be able to boot your old kernel in case something goes wrong. Install the new kernel. Be sure to remember to run LILO before shutting down or you won't be able to boot.

```
mv vmlinuz vmlinuz.old
vi lilo.conf
vi /boot/bootmess.txt
cp -i /usr/src/linux/arch/i386/boot/bzImage
cp -i bzImage /boot/vmlinuz-2.2.17v4l
ln -s /boot/vmlinuz-2.2.17v4l vmlinuz
lilo
```

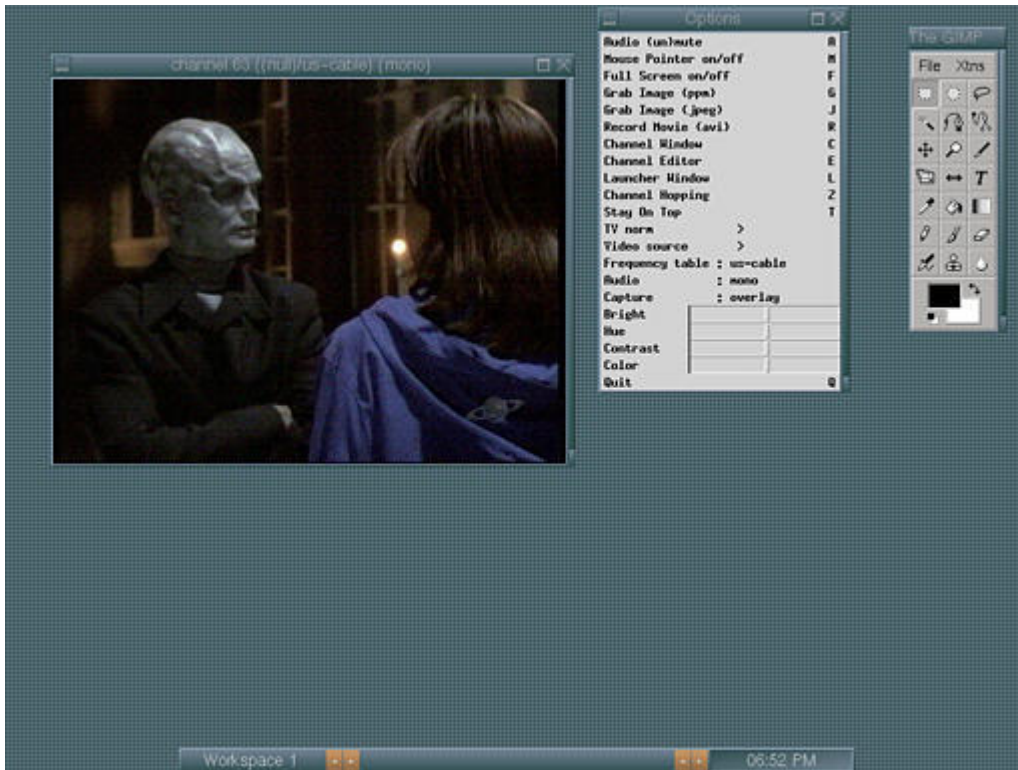
Go back to i2c-5.4 and install our 2.3 driver. We ignore dependency warnings because the old kernel is still running.

```
make install
depmod -a
```

Install the bttv driver:

```
tar xvfz bttv-0.7.51.tar.gz
make
make install
depmod -a
```

Shut down using Ctrl+Alt+Delete and reboot. It boots! We are running our new kernel! A peek at /var/log/messages indicates that bttv is operating okay.



XawTV

Because we installed the current version of V4L we also install the current version of the XawTV viewer program. It won't start because it can't find its X resources. The docs aren't much help. On a hunch we install the old versions of XawTV and fbtv from the Debian CDs, then overwrite it with the new version of XawTV:

```
tar xvfz xaw.3.26.tar.gz
make
make install
```

That works! We right click to get the settings menu, make some adjustments, then use the arrow keys to tune channels. We can now view television on our X desktop.

To recap, the sequence we followed was to install XFree86, configure it, install The GIMP and Netscape, install the updated i2c driver from source, patch the kernel and install Video4Linux and XawTV. Next month we will take a close look at Linux MPEG movie players. We will also install BeOS under LILO and compare using our WinTV card in Linux with BeOS and Windows.



Robin Rowe is a partner in MovieEditor.com, a technology company that creates Internet and broadcast video applications. He has written for *Dr.*

Dobb's Journal, the *C++ Report*, the *C/C++ Users Journal*, *Data Based Advisor* and has had many papers published in conference proceedings. His software designs include a client/server video editing system in use at a Manhattan 24-hour broadcast television news station, Time Warner New York One and associated web site <http://www.ny1.com/>, and an automated television news monitoring system developed for DARPA and the Pentagon. He has taught C++ at two universities and designed video software in Fortune 500, DoD and academic environments. You can reach him at robin.rowe@movieeditor.com.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Turbulent Start for Transmeta

Linley Gwennap

Issue #84, April 2001

Pursuing new markets like webpads may be the way for Transmeta to do more than break even.

Since emerging into the limelight with a widely covered public announcement [see "Linley on Linux" April 2000], Transmeta has had a busy year. The company began shipping its first products, gained several significant design wins and successfully executed an IPO. Although its stock price has since been cut in half, Transmeta still carries an impressive market cap of about \$3 billion US. The company has seen tougher-than-expected competition from Intel in the PC market as well as slow movement in the webpad market. On the other hand, Transmeta recently opened up a third front in the server market. The startup, also famous for being Linus Torvalds' employer, produces the Crusoe chip that is compatible with Intel's processors but uses less power. This accomplishment is due to its unique code-morphing technology that emulates Intel's architecture on simpler VLIW hardware. Initially, Transmeta believed that Intel could not match Crusoe's low-power consumption without developing its own code-morphing technology, which would take years. But Intel fought back quickly.

Without access to code morphing, Intel instead took advantage of its superior manufacturing technology to reduce the power consumption of its chips. Last June, Intel rolled out new processors that operate at lower voltages than Transmeta's IBM-built chips (IBM has manufacturing technology similar to Intel's but reserves it for internal products). Although some studies show that Crusoe still has an edge in power savings, its advantage over Intel's latest chips is small.

Transmeta has also failed to produce any useful performance data for its processor. To be fair, code morphing makes benchmarking difficult; the Crusoe chip actually gets faster if you run a program more often. Testing of initial Crusoe systems found their performance to be roughly that of a mere 300MHz

Pentium III, but these tests ran the benchmark program only once. How fast the chip would run with more “practice” remains unclear, as Transmeta itself still isn't talking.

With its core messages of performance and low power under attack, Transmeta made limited headway in the notebook PC market over the past year. The company placed Crusoe in new ultralight PCs from Sony, Fujitsu, Hitachi and NEC. These systems are aimed mainly at the Japanese market, where space and power savings are at a premium. The leading makers of notebook PCs for the US market—IBM, Toshiba, Compaq and Dell—have yet to roll out any Crusoe-based systems, although all have evaluated Transmeta's technology. As a result, Transmeta's share of the ultralight PC market is perhaps 10% so far. Since that market is just a small fraction of the PC market, Transmeta holds far less than 1% of the total PC processor market. Thus, the company poses little threat to Intel's revenues. Transmeta doesn't need much of Intel's market; however, about 50% of the ultralight PC market would help Transmeta break even.

Investors are hoping for more than breaking even, as Transmeta has lost more than \$200 million since it was founded and will probably lose another \$50 million, even if it reaches its goal of profitability by the end of this year. Investors want some payback. That's where new markets come in. Transmeta's second target market is webpads. These portable devices access web content through a wireless link, showing it on a flat-panel display. Webpads typically use PC-compatible processors, allowing them to run all the plug-ins on the Web but don't need to run Windows; many use Linux instead. The good news is that virtually every webpad announced to date uses Crusoe for its unique combination of low-power consumption, PC compatibility and low price. Companies such as Gateway, Phillips, Hitachi and Frontpath (formerly S3) have selected Crusoe for their webpads.

The bad news: webpad sales have been minimal to date, slowed by prices that are often \$1,000 or more. The most expensive part of these webpads is the flat-panel display, which can cost several hundred dollars alone. However, the recent slowdown in PC sales has reduced demand for flat panels, causing prices to drop. With panel-manufacturing capacity continuing to increase, these prices should fall dramatically and help webpads become more popular in 2001. Still, Transmeta expects webpads to contribute less than 20% of its revenue this year.

Transmeta recently added web servers as a third target market. Although Crusoe is not as powerful as server processors from Intel and others, the low-power chip doesn't need a bulky heat sink or other cooling equipment. This enables vendors to pack several Crusoe chips into the space required by a

single power-hungry Intel chip. Since the task of serving web pages can be easily divided among several processors, Crusoe makes up in numbers what it lacks in individual performance.

So far, only a single small company has endorsed Transmeta's server vision, and Transmeta expects little server revenue in 2001. But this rapidly growing market represents another opportunity for Crusoe, even if Intel can block Transmeta in the PC market. Transmeta hopes PC revenues can tide it over until these new markets begin to flourish. In 2002, we may finally see Transmeta's full business plan in action.



Linley Gwennap (linleyg@linleygroup.com) is the founder and principal analyst of The Linley Group (<http://www.linleygroup.com/>), a technology analysis firm in Mt. View, California.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Focus on Software

David A. Bandel

Issue #84, April 2001

logtool, log_analysis, fwlogwatch and more.

Internets, Intranets, LANs, WANs and more. Frankly, I don't care how a system is connected, be it Ethernet, Token Ring, FDDI, Frame Relay, dial-up PPP, wireless, ham radio, satellite or two cans and a string. If it's connected to something else, even intermittently, it's vulnerable. Recently, Red Hat demonstrated to the world that installing unsecure, vulnerable and, even worse, unnecessary services is a highly security-challenged proposition to the Nth. I don't want to pick on Red Hat; most distributions do similar security-challenged things. But they shouldn't. For my money, no service should be turned on by default, whether the customer asked for a full install or not. Even worse, few distros explain logs and all they offer in their little getting started book. The syslog files (/etc/syslog.conf and the logs themselves) are not "black arts" stuff. They're just boring. Or so we hope. If you have an intruder, or attempted intrusions, these logs can be rather interesting. I've found myself on the edge of my chair as I read through the logs, watching an intrusion and wondering if this wannabe cracker or script kiddie is going to make it in. Okay, so I'm eccentric. But I'm hoping a few offerings centered on system logs might spark a little interest in a bunch of dull log files.

logtool: <http://users.digitex.net/~max/>

The **logtool** utility is another of those small things that sometimes go a long way. All a logtool does is colorize log entries. It makes the date-time stamp one color, originating system another, the facility a third color and the message itself a fourth color. This really breaks out a log entry when you have a large number of them on the screen at one time, making reading entries easier. Requires: glibc.

log_analysis: linux.umbc.edu/~mabzug1/log_analysis.html

The name of this package is a bit of a misnomer. Yes, it does do some log file correlations, but it also shows other things, like currently logged in users (w), filesystem status (df -k), last dump (/etc/dumpdates), the logs. I would say it's more of a system analysis. This won't replace other log file tools that search for anomalies but will give an "executive overview" of a system. Requires: Perl.

fwlogwatch: www.kyb.uni-stuttgart.de/boris/software.shtml

If you're running ipchains, netfilter or a Cisco firewall, this utility can grep your logs and display statistics regarding the traffic passing through (or even just to) your system. You must generate the iptables/ipchains rules for logging (-j LOG in iptables) whatever you want **fwlogwatch** to look for. If you enable netfilter debugging, it's like logging every single packet you see. So I don't recommend that just because of the sheer volume of logging, but it will definitely show you what your system is seeing. Requires: glibc.

MasarLabs System Monitor: www.masarlabs.com/msysm.html

This is another graphical utility used to show various settings and loadings on your system. It is highly configurable and modular, with modules that show apm, clock, CPU, disk status, mail, memory, network status, serial status, swap, network IP, PCMCIA and ppttime. Mix and match in any order you want, in one row across, one row down or in various rows across. Want to just "fill a hole"? Select the empty module. My only complaint is the inability to resize the graphics, which look fine on a screen up to 1024 x 768 but is too small on a screen of 1600 x 1200. Requires: libX11, libXpm, libdl, glibc.

Automated Password Generator and tkapg: <http://www.adel.nursat.kz/>

This password generator can be configured to produce pronounceable passwords as well as totally random "white noise" passwords. **apg** can further check these passwords against a dictionary file. This utility comes as a standard program as well as a daemon that can be run by **inetd** to service requests on the network (this may not be a good idea unless all network traffic is encrypted). The author also provides separately a **tk** utility to access and display generated passwords. These two programs make short work of excuses for bad passwords. Requires: glibc; tkapg also requires Tcl/Tk.

pppstatus: <http://pppstatus.sourceforge.net/>

This utility, designed to be run in a video terminal (VT), shows the status of your PPP connection. All statistics are shown, including IP address and a graphical display of throughput. It's perfect if you have a system that acts as a firewall/

dial-up. Its one drawback is it doesn't have an option to lock the screen when invoked so you can leave it up while unattended. Requires: libncurses, glibc.

Text WINdows Manager: <http://linuz.sns.it/~max/twin/>

Any of you remember the old DOS (DR-DOS or MS-DOS) programs like the Norton Window utility (the name slips my mind) that gave you a window in DOS? How would you like a trip down memory lane? Well TWIN can provide you that trip. It can also provide you with an extremely lightweight term window (or multiple term windows) on one VT. Nice thing is, it also works in X if you're so inclined. I think my laptop just became a non-X piece of hardware. Requires: glibc.

OpenRealty: <http://jonroig.com/freecode/openrealty/>

If you are a realtor, or know any realtors, then this software will be of interest. It claims to be simple enough for a realtor to set up, and I imagine that means techn-eaderthal realtors. Well, that may be a slight exaggeration but not much of one. It will require that someone make adjustments to the index.php page, but, beyond that, this is the simplest package to administer I've seen in a while. I wish realtors had something like this set up the last time I was looking for a house in the States. If you're not in the US, you might need to make some adjustments (including translations), but it would be a trivial undertaking. Requires: web server with MySQL and PHP4, web browser.

Until next month.



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

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Free Beer vs. Free Speech

Rick Lehrbaum

Issue #84, April 2001

Why do developers want to use open-source software, and Linux, in embedded systems?

Throughout 2000, LinuxDevices.com conducted a survey of developers to try to understand their motivations for using Linux in embedded systems and intelligent devices. Some of the most interesting results are in the areas of reasons for wanting to use open-source software and the perceived strengths and advantages of Linux.

You might think the simple answer would be "Because it's free." Not so!

What Do You Value Most about Open Source?

Here, developers were asked to select their first, second and third reasons for using open-source software in embedded applications from these choices:

- So I can add functionality directly within the OS.
- It represents "insurance", even if it's never needed.
- It facilitates debugging and troubleshooting the application.
- It allows a full understanding what's going on inside the OS.
- It lets me immediately fix OS bugs, if they arise.
- It eliminates dependence on a single OS vendor.
- The collaborative open-source development process produces superior software.
- I don't need or want open source.
- Other.

Each of the selected reasons was weighted according to whether it was designated most important (5 points), second most important (3 points), or third most important (1 points). Then, the results were combined and

normalized such that the top reason ended up with a score of 1.0. Figure 1 shows the results.

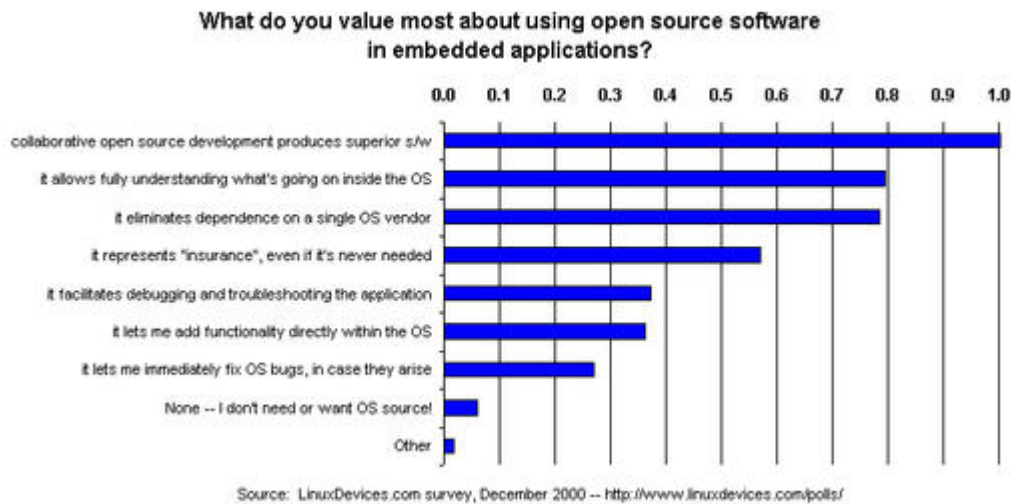


Figure 1. Results of Open-Source Developers Survey

These results are intriguing in several respects. First, the popular notion of programmers hacking away at source code to create custom versions of Linux was not borne out by the survey. Instead, developers place a high value on having source code as a way to avoid being held hostage to proprietary OS providers. Also, having source code makes it much easier to find out what's going on inside the system. Choices like “so I can modify the software” and “so I can fix bugs” did receive a fair number of votes, but in the overall scheme of things, these ended up at the bottom of the list.

Interestingly, the reason that topped the list was “the collaborative open-source development process produces superior software”. What's especially significant about this finding is that it's not something that proprietary software vendors can emulate without fundamentally altering their business models—something they are highly unlikely to do.

Other Reasons for Liking Linux

The survey also asked developers to identify their main reasons for wanting to use Linux in embedded applications. Here, the respondents were asked to check all of the reasons they felt were important from among the following:

- No runtime royalties.
- Source code is available (and free).
- It's not from Microsoft.
- Linux has excellent networking support.
- There are more drivers and tools available.
- Lots of programmers are familiar with Linux.

- Linux is more robust/reliable.
- Other.

Figure 2 shows the results.

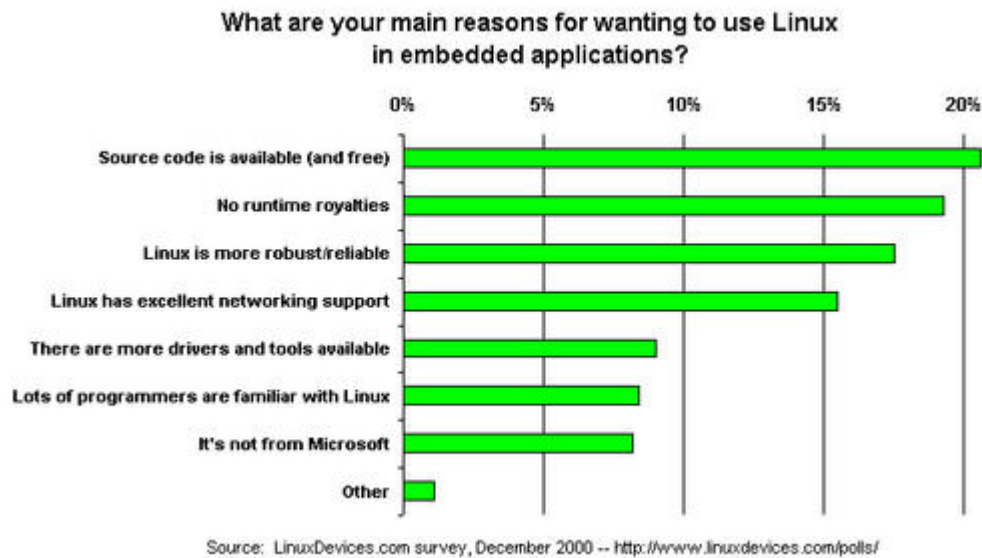


Figure 2. Reasons for Using Linux in Embedded Applications

Free Beer vs. Free Speech

One particularly intriguing result is that, despite the obvious cost-sensitivity of embedded devices, the “free speech” aspect of Linux (i.e., source code is available) edged out “free beer” (i.e., no royalties) as the primary reason developers are looking at embedding Linux in their designs.

To delve a bit deeper into the cost issue, we asked a pair of questions related to costs: “Would you consider paying for Linux development/support services?” and “Would you consider paying per-unit royalties?” The results appear in Figures 3 and 4. (Note: the second question was added more recently, so the results shown here are based on a relatively small sample of data.)

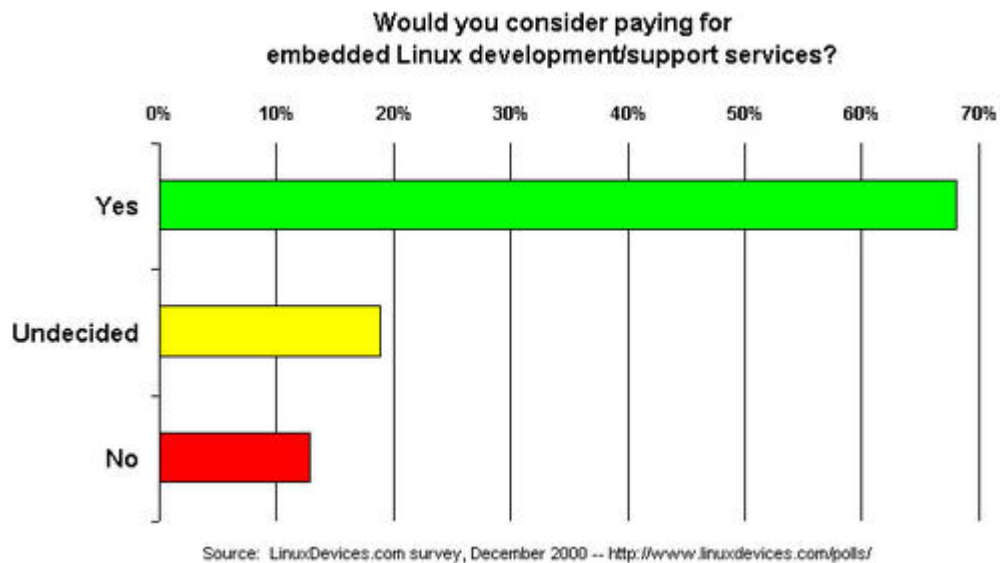


Figure 3. Would you pay for Linux development/support services?

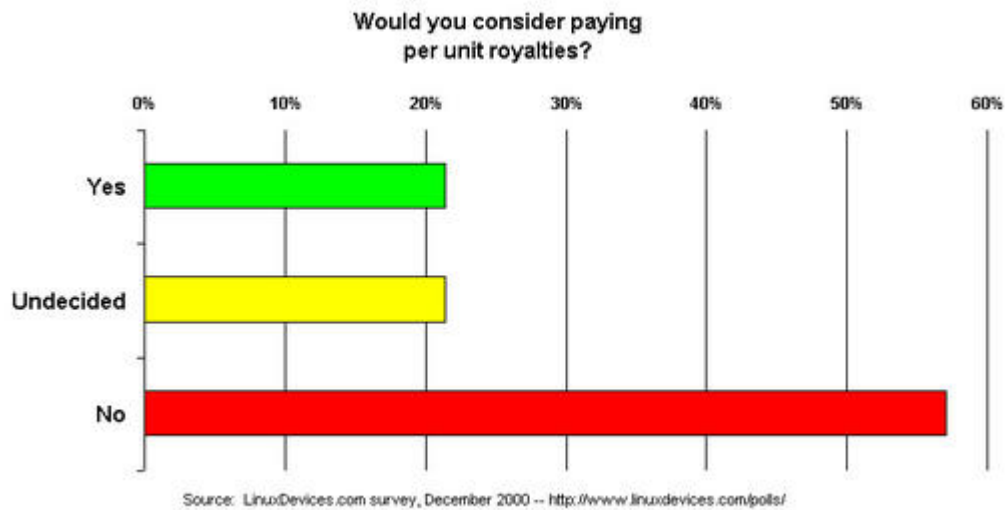


Figure 4. Would you pay per-unit royalties?

What we learn from the results is that while embedded developers are indeed prepared to invest money in outside services and support for embedded Linux (68% said yes and only 13% said no), the numbers are nearly flip-flopped when the question is willingness to pay royalties (51% said no and only 21% said yes).

I suspect some of the suppliers of embedded Linux software and services will find these results "interesting"--to say the least!

Please vote in the new 2001 Embedded Linux Market Survey, even if you already participated in last year's survey. Go to www.linuxdevices.com/cgi-bin/survey/survey.cgi.



Rick Lehrbaum (rick@linuxdevices.com) created the LinuxDevices.com “embedded Linux portal”, which recently became part of the ZDNet Linux Resource Center. Rick has worked in the field of embedded systems since 1979. He cofounded Ampro Computers, founded the PC/104 Consortium and was instrumental in launching the Embedded Linux Consortium.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

The New Vernacular

Doc Searls

Issue #84, April 2001

“Habitability is the characteristic of source code that enables programmers...coming to the code later in its life, to understand its construction and intentions and to change it comfortably and confidently....Habitability makes a place livable, like home.” --Richard Gabriel

We can't talk about software without borrowing the language of construction. We call ourselves builders, designers, architects and engineers. We speak of structures, spaces, objects, frameworks, levels, partitions, elements, components and platforms. We assemble developing software into builds. Finished work takes on the rhetoric of real estate when we build sites with addresses and locations.

So what's going on here, beyond promiscuous linguistic leverage? For a while I've insisted that it means the software industry is maturing into a construction industry—one defined by skilled and reputable practitioners rather than by the sole-source suppliers we call “platform providers”. In a mature software industry, Microsoft will be no more or less important than, say, Georgia Pacific or Kaufman & Broad.

In the construction industry, open source is standard. When building materials and methods aren't secret, there's more to talk about, more people involved in the conversation and more people contributing to the improvement of those materials and methods. This is a bit more low-falutin' than the peer review process that Eric Raymond talks about, but it's the same thing. Peer review happens constantly in every living, growing industry, at every level. Consider *Architectural Graphic Standards*, a sourcebook for design and construction details first published in 1932. More than half its contents are new or revised every seven years.

The software industry is just a few decades old while construction is as old as civilization itself. This suggests we might have a few things to learn from the

senior industry. Richard Gabriel seems to agree. In *Patterns of Software* (Oxford Paperbacks, 1996) he extols the virtues of building software for “habitability”. His structural ideal is a New England farmhouse:

The result is rambling, but each part is well-suited to its needs, each part fits well with the others....The inhabitants are able to modify their environment because each part is built according to the familiar patterns of design, use and construction and because those patterns contain the seeds for piecemeal growth.

In *How Buildings Learn* (Penguin Books, 1994), Stewart Brand calls the New England farmhouse a perfect example of “vernacular” construction. He writes:

What gets passed from building to building via builders and users is informal and casual and astute. At least it is when the surrounding culture is coherent enough to embrace generations of experience.

Vernacular is a term borrowed since the 1850s by architectural historians from linguists, who used it to mean the native language of a region....It means common in all three senses of the word—widespread, ordinary and beneath notice.

In terms of architecture, vernacular buildings are seen as the opposite of whatever is academic, high style, polite. Vernacular is everything not designed by professional architects—in other words, most of the world's buildings....Vernacular building traditions have the attention span to incorporate generational knowledge about long-term problems such as maintaining and growing a building over time. High-style architecture likes to solve old problems in new ways, which is a formula for disaster....

Vernacular buildings evolve. As generations of new buildings imitate the best of mature buildings, they increase in sophistication while retaining simplicity.

Is UNIX vernacular? Here's what Neal Stephenson says in his book *In the Beginning Was the Command Line* (Morrow, William & Co., 1999):

The filesystems of UNIX machines all have the same general structure. On your flimsy operating systems, you can create directories (folders) and give them names like Frodo or My Stuff and put them pretty much anywhere you like. But under UNIX the highest level—the root—of the filesystem is always designated with the single character “/” and it always contains the same set of top-level directories:

```
/usr/etc/var/bin/proc/boot/home/root/sbin/dev/lib/  
tmp
```

and each of these directories typically has its own distinct structure of subdirectories. Note the obsessive use of abbreviations and avoidance of capital letters; this is a system invented by people to whom repetitive stress disorder is what black lung is to miners. Long names get worn down to three-letter nubbins like stones smoothed by a river.

It is this sort of acculturation that gives UNIX hackers their confidence in the system, and the attitude of calm, unshakable, annoying superiority. Windows95 and MacOS are products, contrived by engineers in the service of specific companies. UNIX, by contrast, is not so much a product as it is a painstakingly compiled oral history of the hacker subculture.

Vernacular architecture is the opposite of what Brand calls “Magazine Architecture”, architecture as art, rather than as craft. Here's the difference, according to Henry Glassie: “If a pleasure-giving function predominates, it is called art; if a practical function predominates, it is called craft.” It's hard to imagine anything more crafty and practical than a command-line interface.

Stewart Brand finds a lot of craft in what he calls “Low Road” buildings, which tend to be “low visibility, low-rent, no-style”. He says, “Most of the world's work is done in Low Road buildings...and even in rich societies the most inventive creativity, especially youthful creativity, will be found in Low Road buildings, taking full advantage of the license to try things.”

So, it's hardly a coincidence that Brand's ideal Low Road building is MIT's Building 20, where “The Tech Model Railroad Club on the third floor, E Wing, was the source in the early 1960s of most of the first generation of computer hackers who set in motion a series of computer technology revolutions (still in progress).”

We tend to think of revolutions as rapid, but the revolution that started in MIT's Building 20 predates Moore's Law, which was first published in 1965. It also predates the software industry by almost a generation. Any way you look at it, the hacker subculture has served as a cultural foundation for computing for a very long time. It has also persisted in a comparatively stable form while commercial fashions and revolutions have come and gone, again and again. I'm not saying UNIX does not have a commercial side; just that its cultural foundations are deeper than commerce.

In *The Clock of the Long Now* (Basic Books, 1999), Stewart Brand sorts civilization into six layers that change at different rates over time.

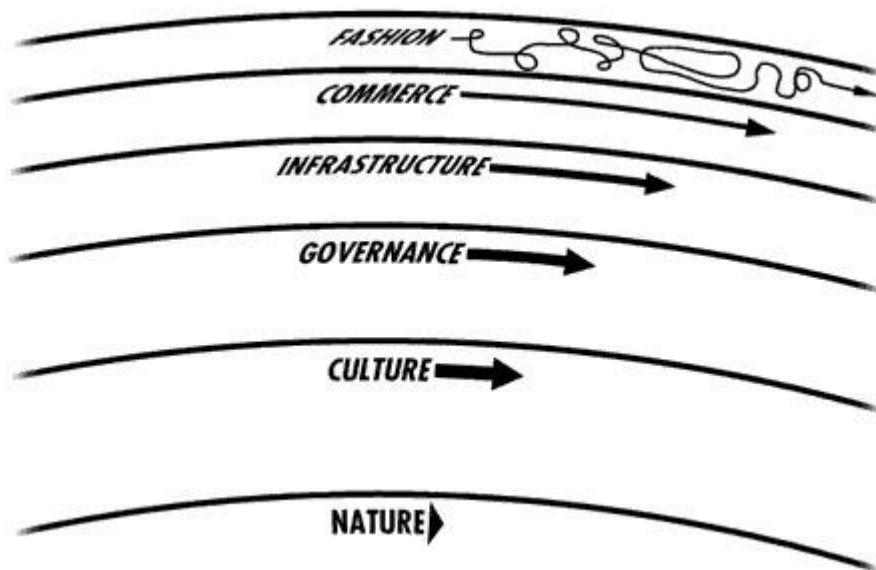


Figure 1. Civilization's Six Layers

From bottom to top they range from slow to fast. "The fast layers innovate", he says. "The slow layers stabilize. The whole combines learning with continuity."

This puts both hacker culture and software commerce into a fresh and interesting perspective. Positioned one level above nature, hacker culture is often concerned with the natural qualities of software. The GNU Manifesto, for example, says free software is "just like air". Going up one level to governance, we find the hacker culture's obsessive concern with license agreements. These agreements manifest one layer up in infrastructure: software and protocols (a form of both governance and cultural agreement) that increases in value as it approaches ubiquity.

I don't think it's a stretch to say that hacker culture has a natural understanding of what makes software truly valuable. We see manifest in the Internet, which possesses three almost natural qualities: nobody owns it, everybody can use it and anybody can improve it.

These are built into the Net—and into free software development tools and licenses—for everybody and for fundamentally social reasons. That's why the GNU Project says, "people should be free to use software in all the ways that are socially useful." This is embodied not only in the GNU tools and other free software, but in Linux, BIND, TCP/IP, sendmail, Apache, Jabber and SOAP—all of which (for the most part) nobody owns, everybody can use and anybody can improve.

These are values that do not—and cannot—come from business. They are not commercial values. However, the layer of software civilization we call commerce depends on the social infrastructure that has grown up out of the

hacker culture and UNIX in general. This obviously includes the Net but also lots of other practical and ubiquitous stuff that's free as air.

It is important to understand how these layers relate because a lot of misunderstandings and bad decisions get made when, say, commercial interests attempt to impose self-interested infrastructure, or when governance attempts to socialize commerce.

Michael Polanyi says “comprehensive entities [such as civilization] are logical combinations of levels of reality”, and there are principles—boundary conditions—by which each lower level supplies conditions on which upper levels depend. Phil Mullins puts it this way: “A lower level imposes restrictions within which a higher level can come to operate; the lower level establishes boundaries but leaves open possibilities. The higher level cannot be exhaustively described in terms of the lower level...no level can gain control over its own boundary conditions and hence cannot bring into existence a higher level.” This is why the Net didn't create e-commerce, yet e-commerce depends utterly on the Net.

So, while the Free Software movement might appear anticommercial to many in the purely commercial sector of the software industry, in fact, it just isn't very interested in what happens at the commercial level, and is even less interested in what happens way up on the fashion level. There are more enduring, more purely cultural, more natural concerns.

When we look at the boundary conditions that flank infrastructure, we also see why software companies run into problems when they try to civilize their industry from the top down. Microsoft may have been successful at this but only temporarily. These days all of us, including many at Microsoft, are starting to feel software getting civilized from the bottom up, thanks to the Net and the culture that built it.

We see progress today in protocols like SOAP and XML/RPC, which were developed to support publishing on the Web (so, for example, you can write simultaneously for multiple sites other than your own). They're wide open and available to everybody. Dave Winer and his company, Userland, were involved in both efforts. So were Microsoft and Developmentor, a training and education company. When I asked Dave how the efforts went, he said, “Working with Microsoft and Developmentor on both projects was the best collaborative development experience I have ever had.”

Microsoft comes from personal computing and always will. But it has to thrive in a world where computing is more and more social. And social computing has been built into UNIX since the beginning.

For software businesses, Craig Burton says, “the real challenge is to create ubiquitous infrastructure while generating shareholder value”. It isn't easy, but it helps to do two things. One is comply with your own engineers, who are busy making infrastructure whether you like it or not. That's what IBM did when it embraced Linux: a phenomenon that was growing like wildfire inside its own enterprise. The other is to recognize what works over the long term. The Long Now Foundation proposes seven guidelines for a long-lived, long-valuable institution:

1. Serve the long view (and the long viewer).
2. Foster responsibility.
3. Reward patience.
4. Mind mythic depth.
5. Ally with competition.
6. Take no sides.
7. Leverage longevity.

Institutions do learn—even the ones obsessed with art and fashion. Take Apple. Perhaps the most significant item among Apple's many rollouts early this year was the farthest from fashion and, therefore, the least reported. It happened down at the level of governance. After seventy thousand hackers jumped in to improve Apple's BSD-derived Darwin OS, Apple responded by adjusting its source code license in the direction of “the nature of things”. The new license may not be a clone of the GPL but it's a whole lot closer. Chris Bourdon, the Product Manager for OS X, told me, “You do take Darwin and do anything you like. It's there for everybody.”

Of course it is. It's UNIX.

Doc Searls is senior editor of *Linux Journal* and coauthor of *The Cluetrain Manifesto*.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

Descent3 for Linux

Neil Doane

Issue #84, April 2001

Kudos to the fine folks at Outrage and Loki for getting the latest Descent into our sweaty little hands.



- Manufacturer: Loki Entertainment Software
- E-Mail: sales@lokigames.com
- URL: <http://www.lokigames.com/>
- Price: \$24.95 US
- Reviewer: J. Neil Doane

This month's review is of yet another great Loki game. Honest folks, we're not playing favorites here; Loki is just so forthcoming and has so many store-shelf titles out there right now it's inevitable we're going to have a bit of a Loki deluge from time to time. We do have some other great non-Loki titles in the pipe and will be spending some time on Linux's noncommercial gaming as well, we promise!

For now, I meant it when I said that this is a great game; this game rocks. *Descent3* is the first *Descent* version to make it to Linux, and it looks like we're

getting the best installment in the series so far. Kudos to the fine folks at Outrage and Loki for getting the latest *Descent* into our sweaty little hands. Some of us still remember the awe we felt when we played the original *Descent* for the first time, twisting through subterranean mineshafts, floating weightlessly in space and the dizzyingly true 3-D maneuvering possible from *Descent's* six-way motion control system. The original *Descent's* biggest issues were its repetitive nature and simple plotlines. Even its successor, *Descent2*, saw little improvement on these fronts, mainly adding support for new hardware, some improved game play and some new weapons to enjoy. *Descent3* finally gets it right; this game has both an interesting story and unique levels with diverse, multiple goals, yet manages it all without straying too much from the basic feel players have come to expect from *Descent-series* games.

The Story

The story is not too complex, but it's entertaining and well written (at last). The visually stunning introductory movie and the first premission cut-scene movie are each five full minutes of completely 3-D modeled plot development. *Descent3* essentially takes up right where its predecessor left off: you're unconscious and adrift in space after the explosion that rocked your world at the end of the *Descent2*. At the last moment, as you and your ship plunge helplessly to your doom into a solar corona, you're rescued by some new friends. They reveal that your old employer, the PTMC, has been doing some rather nasty stuff, such as developing weird nanotech viruses designed to turn mining robots into bloodthirsty military tools. Worse, they're apparently responsible for sabotaging your ship and attempting to kill you! (Okay, it isn't Herman Melville, but it's a good enough story to keep you interested.) The rest of the game is basically about flying around in your new spaceships (you eventually get three), attempting to subvert the PTMC's plans, rescuing key detectors involved in the virus project, exposing the PTMC's evilness to the world and, more or less, opening up a proverbial can of Material Defender-style whoop-ass on your ex-employer and would-be assassin. The story line unfolds fluidly through the use of 3-D modeled cut-scenes and radio transmissions. It is actually quite compelling at times and has some unexpected twists. Though *Descent3* is still a game that could be played with no prior exposure to the plot, it is now good enough to be engaging and fun to watch unfold.

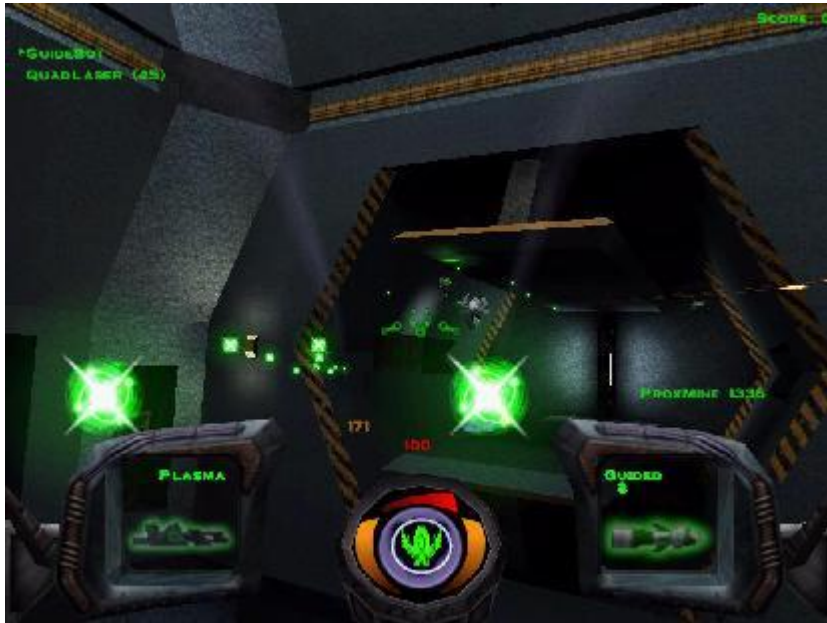


Figure 1. The Plasma Cannon: when you absolutely, positively gotta wipe out every PTMC robot in the room.

Game Play

Combining the ease of movement in a first-person shooter with the directional freedom of a spaceship is something *Descent* has excelled at from the beginning. The task is not without cost, however, and many players find that keyboard-only control is extremely less effective than using a joystick. There are three axes of motion and three directions of travel to deal with, so coordinating the full use of all motion controls while navigating and fighting unfriendly robots in tight underground passageways is often a rough proposition.

The game itself is a series of missions, 15 in all, each with its own unique objectives. Though the “find keys to open doors to get to goal” motif is still present to at least some degree in most levels, most have several challenging sub-goals to reach, as well as occasional puzzles that must be solved in-game, some of which can be quite challenging. Your enemy's artificial intelligence is pretty high-quality stuff and is a major improvement from past *Descent* versions. You'll find that they attack in groups and use strategy (often falling back when in danger to regroup), communicate with each other and announce your presence to other robots in your general area (which typically brings in the cavalry to cause you more headaches, some of which will sneak around behind corners and lurk in waiting for you to pass by, while others will get right in your face, or worse, sneak up right behind you). Thankfully, *Descent3* comes with ten brand-new weapons for you to use on these little twerps. Another of Outrage's triumphs, their new “Fusion” rendering engine, allows another new facet to *Descent's* levels: outdoor missions. No longer are you continuously trapped in the claustrophobic depths of some darkened mineshaft; now you can soar in the skies above the extraterrestrial surface. The underground levels themselves

are well designed, varied in setting, often incredibly complex in layout and generally look spectacular. All the levels are rather gargantuan and complex; with only 15 total levels in the single-player mode, each tends to consume a significant amount of time and effort to complete. Luckily, in *Descent3*, you're not alone down there....

Your little buddy from *Descent2* is back! *Descent3* also gives its pilots the ability to use a "guidebot". Guidebots are small robots who accompany you down in the tunnels, are carried by your ship and act as roving robotic assistants. In times of need, you can launch your guidebot and ask him to help you do any number of things, from leading the way out of the seemingly endless maze of tunnels in order to help you find your next goal (patiently waiting for you to follow it along) to putting out fires or showing you the way to the next enemy robot. Occasionally, the guidebot gets a bit confused or forgets the next goal, and from time to time, he gets stuck in weird places, but he is often an invaluable sidekick.

Descent3's audio/video quality is truly inspired: the graphics are stunning; the environment can be manipulated and destroyed; the special effects sounds are realistic and crystal clear (and do much to provide the perfect level of ambient noise in the tunnels) and even the background music sounds not only good enough to leave turned on, but good enough to turn up. My only issue with *Descent3's* OpenAL-accelerated sound is that it made me wish I had larger speakers and more forgiving neighbors.

Multiplayer

Descent3's multiplayer support is, well, phenomenal. You have nine types of multiplayer games to choose from, ranging from the favorites like "Capture the Flag" and "Deathmatch" (Anarchy mode) to the much less frequently seen (but often missed) "Cooperative" mode. The "Monster Ball" mode is kind of like a soccer game with spaceships. You can play in TCP/IP-based mode for direct connections, or if you're looking for a game, you can connect to Outrage's Parallax Online (PXO) gaming network and join a game in progress. The *descent3* binary can also be run by itself as a standalone game server for the truly hardcore who want to host their own dedicated *Descent3* server.



Figure 2. Take down the HUD for those fights when maximum visibility is critical.

Linux Notes

Descent3's installation footprint can get hefty (up to 1GB), but Loki's installer includes options for a much smaller install, mainly done by leaving the movies on CD. The game's hardware requirements are pretty low for the intensity of game play offered. Loki recommends only a 300MHz CPU (and even lists 200MHz as the bare minimum) and 64MB of RAM. One pitfall that Loki doesn't mention is that if you leave the movies on CD, you might want to consider a slightly faster CD-ROM than the 6x minimum recommended by Loki; my older 8x drive was choking and coughing in several of the more intense movie scenes. Software requirements include the usual Loki list: Linux kernel 2.2.x and Glibc 2.1 and, of course, for network play you'll need TCP/IP support and an internet or LAN connection.

One item on *Descent3*'s minimum equipment list is a 3-D accelerator card. Several card options are supported and have been tested, including most of the more popular 3-Dfx Voodoo series (by way of a nice dedicated Glide renderer), the Matrox G200/G400-series cards (by way of Utah-GLX and Mesa drivers) and NVIDIA cards supported by NVIDIA's OpenGL-endowed X server (in-game testing, this reviewer's GeForce GTS worked flawlessly). For a complete list of cards and drivers that have been tested with *Descent3*, it's probably best to check <http://www.lokigames.com/> for the latest news.

Optional equipment for *Descent3* includes the Rock'n Ride gaming chair available from <http://www.rocknride.com/> (the coolness of which really defies description). Loki has even been kind enough to include detailed instructions on how to use the new *Descent3: Mercenary* expansion pack for Windows with the Linux version of *Descent3*. Very sweet.

Summary

Again, what can I say? This game is very entertaining, very well written, very well executed and extremely fun to play. The graphics are exceptional, the sound effects are clear and high-quality—even the music is great. The basic movement controls may be a bit complex but nothing that shouldn't be expected for the amount of control you have over your ship. The general game controls are simple, easy to operate and logically arranged. The multiplayer action is second to none, and the 3-D support is trouble-free. A truly excellent game; for the first-person shooter genre, this is one of the coolest titles out there.

[The Good/The Bad](#)

[Linux Game Update](#)



J. Neil Doane (caine@valinux.com) is a professional services engineer with VA Linux Systems and an Indiana escapee. Between prolonged spasms of rabid geekness, random hardware scavenging and video gaming, he is a pilot, a guitarist and a very poor snowboarder.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

LinuxPPC Goes Nonprofit

Leslie Proctor

Issue #84, April 2001

On LinuxPPC, LinuxFund.org and more.

This has been a big month in the .org world. I think some of the most interesting news (in full below) is the migration of LinuxPPC from a for-profit company to a nonprofit .org. LinuxPPC was originally conceived as a nonprofit, but the market dictated for profit, and now the timing is right to make the move back to its original vision. The principals will no longer need to worry about path-to-profitability issues.

I think it begs the question that has been asked hundreds of times: can for-profit companies succeed in Linux? The answer lies in how well for-profit companies interface with the .org world. Those that have very active community efforts seem to be thriving, while many who pop on to the scene without an understanding of what the Linux community is really all about are swimming upstream. A solid community outreach program should be an integral part of any project's strategy. The need to be hooked into the .org world has never been stronger.

Profile—LinuxFund.org

LinuxFund.org (www.linuxfund.org) raises capital to fund high-quality Linux projects by issuing an affinity MasterCard. The project is headed by Benjamin Cox who describes himself as "the financial world's version of a hacker". In eighteen months, LinuxFund.org has issued more than 5,000 credit cards and looks to fund two or three projects per month.

Any project in the world can apply, but they must be completely open source and help the development and direction of open-source software. The organization is currently funding fewer projects per year at \$3,000 per project but are looking to spread the fund wider by offering more \$1,000 grants. They

are actively seeking projects to endow. They'd also like to eventually be a source to find information on interesting open-source projects.

LinuxFund.org does not solicit or accept donations. The sole source of their funding comes from the affinity credit card. Cox feels that the card acts as an ambassador for Linux. Nearly every time he uses it, someone comments on the card, Cox said.

LinuxFund.org is actively looking for projects to fund. Submissions can be posted on-line at the organization's web site, <http://www.linuxfund.org/>. They are also always in the market for web development help.

News Briefs

LinuxPPC Becomes a Nonprofit Organization: LinuxPPC (www.linuxppc.com) announced plans to become a nonprofit organization (NPO). The NPO will focus on the development and promotion of the Linux operating system on the PowerPC processor. Jeff Carr, LinuxPPC's founder, originally registered the linuxppc.org domain with the intention of creating a not-for-profit organization. However, it was officially created as a for-profit corporation because it was less difficult and less expensive than creating an NPO.

The organization states that since many of the ideas and principles of nonprofit organizations perfectly match the ideas and principles of the free software movement, the company calls the move "a natural step in LinuxPPC's evolution". The transition is expected to take several months and will not affect the services offered by the organization. LinuxPPC is headquartered in Waukesha, Wisconsin.

Free Software Foundation (FSF) Award for the Advancement of Free Software: The Free Software Foundation (<http://www.gnu.org/>) bestowed its third award for the Advancement of Free Software to VA Linux employee, Brian Paul. Brian was given the award for his work on the Mesa 3-D graphics library. The award, a one-of-a-kind handmade quilt, was presented by Richard Stallman at the Linux Expo in Paris.

Brian was selected from three finalists who were all honored at the award ceremony. The other finalists were Donald Becker, nominated for his network device drivers for the GNU/Linux system and for the Beowulf Project; and Patrick Lenz for his work on freshmeat.net, a vital site for news and information on free software.

GNOME Foundation: The GNOME Foundation announced that RedFlag Software Co., Ltd. has joined the foundation. RedFlag will spearhead an effort to localize GNOME into Simplified Chinese.

“With the localization of GNOME, businesses and individuals in China can enjoy the power, cost-effectiveness and ease of use of free software,” said Havoc Pennington, GNOME Foundation board chair. “In addition, talented programmers in China will be able to utilize the modern architecture and design of GNOME for application development.”

RedFlag will lead GNOME Foundation efforts in China, focusing on education, support and marketing. It will also join the volunteer efforts of more than 500 of the world's most talented software designers and programmers who are currently working on GNOME or GNOME-compatible programs.



Leslie Proctor is an active participant in the .org community and has acted as a consultant for a number of .org organizations around the world. Send mail to lesproctor@yahoo.com.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

Appgen Moneydance 3.0

Joseph Cheek

Issue #84, April 2001

Moneydance isn't a bad product. It's got a few glitches but nothing major.

- Manufacturer: Appgen Business Software, Inc.
- E-mail: sales@appgen.com
- URL: <http://www.appgen.com/>
- Price: \$39.95 US (free Beta 3.0 tested)
- Reviewer: Joseph Cheek

The brochure said “The first personal finance software is written completely in Java”, and also “Import and export QIF files, search detailed transactions, reconcile checkbook balance easily, print customized and standard reports and 3-D graphs, do XML imports and exports—just load the enclosed CD”.

I have wanted to track my personal finances on Linux, and thought this might be a good way to do it. I had tried GnuCash [see page 122 for details about this alternative] but found it not feature-rich enough for my tastes. This CD, which says it runs “on virtually any operating system from Linux, FreeBSD, MacOS X or OS/2, Sun Solaris, MacOS and Windows”, sounded good. I popped it in the CD-ROM drive to give it a whirl.

The CD label clearly states “Prerelease version, not for sale”. I'm not sure how prerelease this is, but I anticipate this will be very similar to the release version. The screen shows, “Moneydance 3.0-BETA (153)”.

Hmm. The CD root directory has several files that appear to be installers: `install.bin`, `install.exe` and `install.sh`, as well as `Setup.kdeInk` files. The `Setup.kdeInk` file looked promising, but I had a hard time clicking on it—no icon. Once I got it working, I was greeted with a text window with this message:

```
Mon Jan 22 15:21:47 PST 2001
Welcome to Moneydance install press return to continue
```

I pressed the Enter key.

```
using JVM: /mnt/cdrom/jvms/Linux_i386_jre_1.1.7v3.tar
Where would you like to install Moneydance?
(We recommend /opt/moneydance)
```

Ahh, this looked like a good spot to install. I pressed Enter again.

```
Where would you like to install Moneydance?
(We recommend /opt/moneydance)
```

Okay, I have to type it out. Fine, **/opt/moneydance**.

And the text window disappeared. I had expected something like, "Installing Moneydance, please wait", or "Install completed, press Enter", or "Install failed, you don't have enough permissions" or something. I got nothing.

Looking through Setup.kdeInk, I saw that it called the install.sh I had found on the CD but failed to call it with root permissions. Note to Appgen: adding X-KDE-SubstituteUID=true and X-KDE-Username=root will tell KDE2 to prompt for the root password and su before calling the script. Barring that, please add an error handler with a pause at the end so we can see the error messages the script creates.

So the install procedure becomes: pull up our xterm, **su** to root, and enter **/mnt/cdrom/install.sh**. Now Moneydance installed fine. Running it from a command prompt, I am shown:

```
This is an unregistered version of Moneydance.
To register please visit the following URL:
moneydance.net/purchase
Error: unable to initialize 1.1 https:
java.lang.ClassNotFoundException:
com.seanreilly.apps.moneydance.controller
HTTPS protocol handler NOT initialized!
on-line banking will not work.
```

On-line banking would have been cool. Oh, well. Next I am asked for my currency and am offered a choice of US, Australian or Canadian dollar; Brazilian Real; German Mark; Danish Krone; Euro; French Franc; British Pound; Italian Lira or Japanese Yen. The documentation sheet I was given adds that "Moneydance has been translated into French, German, Spanish and Brazilian Portuguese". This is certainly an international program.

The main window popped up (see Figure 1). It's pretty unencumbered. The current date and time, along with the current month's calendar, are at the top of the screen, and next come summaries of my accounts (none yet) with a big goose egg for my personal worth.

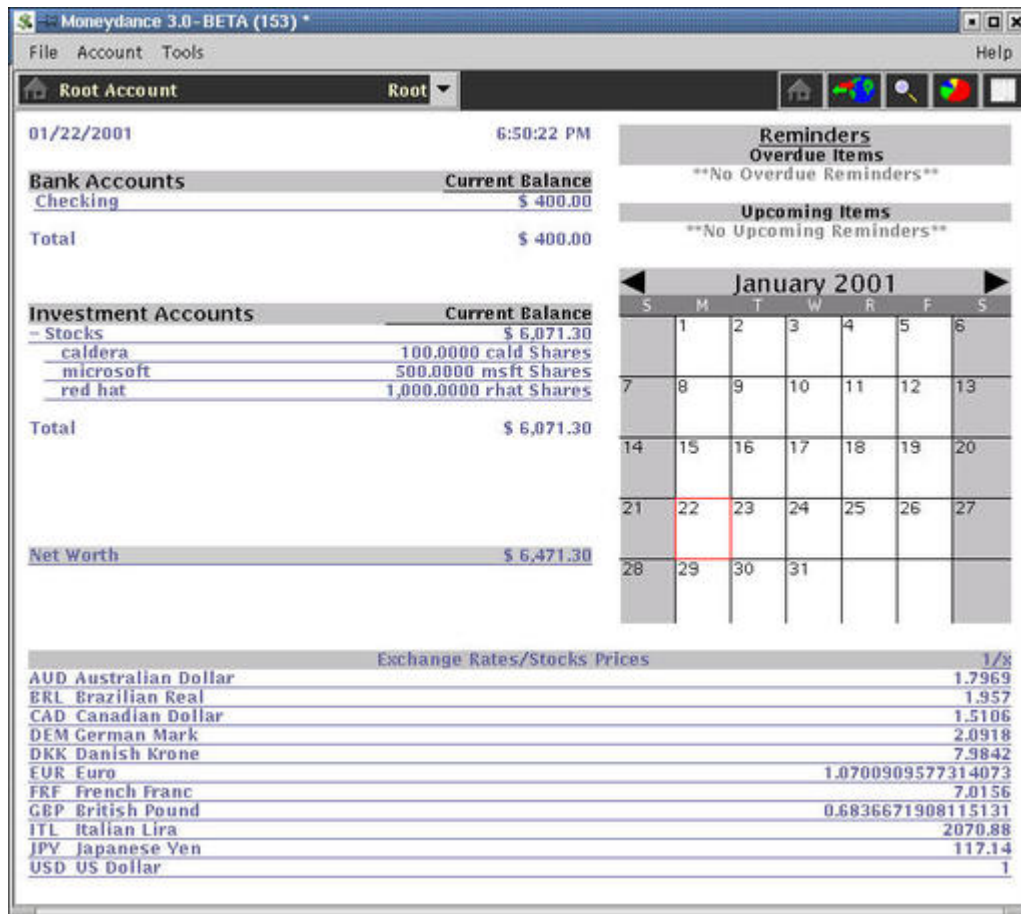


Figure 1. The Moneydance Main Screen

At the bottom of the screen are currency exchange rates for the 11 currencies Moneydance supports. I found a button that refreshes these from data on the Internet. The heading told me it would list stock prices too, if I had any. Now to find out how to add them.

Time to create some accounts. The account menu (see Figure 2) lets me choose a bank, credit card, investment, asset, liability, loan, expense or income account. I chose bank and clicked on Next. I was prompted for an account name, bank name, account and routing numbers, an initial balance and currency type.

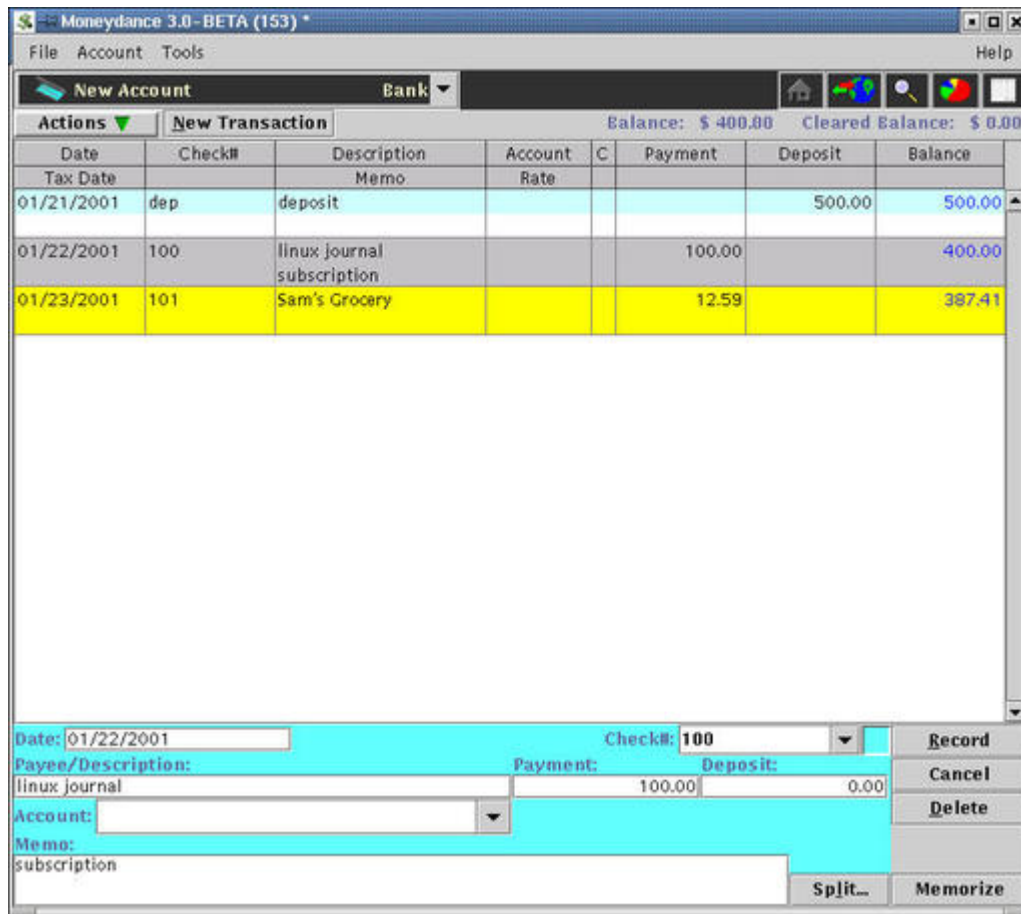


Figure 2. The Account Detail Screen

Should I give this account a default category? The only category I had was listed with a diamond and labeled Expense. Sure, why not. Should this account be a child of another account? It gave "Root Account" as my only option. Okay. The last option was labeled "Check# items" and showed:

```
CkCrd
Dep
Trn
ATM
{Print}
<Next Check#>
```

I wasn't sure what the significance of this was. I could add new entries, shuffle the order of entries and delete entries from this list. I chose not to do any of these.

Back on the main screen (the "Home Page", it tells me) I have a checking account with \$0 in it. Clicking on it brings me to a detailed transaction window that lets me enter new transactions, reconcile my account, print checks, and set up on-line banking and bill payment (neither of which work yet). I can get graphs of my transactions also, but first I have to enter some.

Let's see...if I start with \$100, I can pay myself \$50, give \$25 to *Linux Journal* and still have \$25 left over. Can I get a graph now? Yes, of Expenses, Income,

Income and Expenses or Account Balance, grouped by Day, Week, Month or Year. I clicked "Account Balance", "Group By Day" and "Done". Nothing. Try again—"Graph", "Account Balance", "Group By Day"...aha, the Done button is mislabeled. It should read "Cancel this action, I really didn't want to use all of the data I had entered". The "Generate Graph" button means "show me the graph".

The graph is a very pretty but a slightly unsettling "V" in a large window. While the V was very nice, and told me **Account Balance: checking ??details??** whenever I hovered the mouse pointer over it, the high balance of \$150 (after a few more transactions) was at the extreme top of the screen, and the low balance of \$111.98 was at the extreme bottom. In vain I tried to find a way to adjust the scale so that I could avoid making it look like I was dead broke with \$111.98 in the bank. I settled with knowing that a dip of the graph to the very bottom of the screen doesn't mean I'm broke. Phew.

I could print or save this graph and toggle 3-D mode on and off. The print command appears to print standard PostScript to a printer (such as lp) in portrait or landscape mode with letter, legal, executive or A4 formats. Hmm, printing from Moneydance doesn't work to my remote LPRng printer. Ick.

I went back to the main window—er, "Home Page"—and checked the overview. I now had a bank account with \$111.98! After reloading the exchange rates (they didn't change; it must not be real time) I decided to add a stock account. Account—>New—>Investment—>Next. To create an investment account, it wants the bank name and account number. Was this right? I wanted to track stocks, not a bank account. Cancel. Perhaps Account—>New—>Asset—>Next. Nope, that's not right. I guess investment was it.

Creating an investment account and clicking on it, showed me tabs labeled "Portfolio View", "Register" and "Securities Detail". This was what I wanted. On the right side of the window was an "Add Security" button. Enter the name, the symbol, the type (CD, Bond, Mutual or Stock), the price, the broker, the phone number and comments. I decided to track MSFT and RHAT.

In the securities detail screen I could buy, sell, track splits and show a history of the stock. I could download data points from Yahoo or add and remove them myself. Downloading from Yahoo, I saw that over the past few months RHAT has closed between 5 and 9.2. Performance was shown on a three-month scale with no option to adjust; the data points were no more granular than one every day or two. Good thing I'm not a day trader.

I told Moneydance that I had bought 100 shares of MSFT for \$50 and sold them the same day for \$70. Not a bad profit. It didn't even ask me which account I

used to pay for them. They closed at \$61, so after a few transactions I was a few thousand dollars richer. Wow, if only I could do that in real life.

The news button didn't show me anything; it would have been nice, but I really just wanted to create virtual wealth anyway. Back at the home page, I asked for another update on exchange rates and stock prices. Hmm, stock prices don't show up in the Stock Prices section. Clicking on MSFT again I notice it's gone down to \$60.125. Moneydance's data points aren't granular enough to be good for making investment decisions; it's really just for tracking how well you did in the aftermath.

I now had a virtual net worth of \$4500, due to some creative data entry and a good look at Moneydance. The only other option that interested me was the Budget Manager from the Tools menu. I could choose any expense account (for some reason, I now had three expense accounts, all labeled with a single diamond) and Moneydance would create a budget based on past transactions. While that is a handy feature, I couldn't find a way to edit the budget—I could only accept what it presented me with, and I couldn't create new budget categories—and the only time it was used was in one report, the Budget report.

What I really wanted was a way to assign transactions from my checking account to different categories and have the computer tell me how close I was getting to my monthly allotment of funds. Not so. Appgen, consider this a feature request for the next version please.

My thoughts: Moneydance isn't a bad product. It's got a few glitches but nothing major. It's stable and works well for what it's meant to do. I like it better than GnuCash, but GnuCash is free, and Moneydance isn't. Would I buy Moneydance? Not without a budget tracker.

The Good/The Bad



Joseph Cheek is a senior Linux consultant with the Professional Services Group in LinuxCare, Inc.'s Seattle office. He spends most of his free time playing with his wife and two daughters and working on Redmond Linux. He can be reached at joseph@linuxcare.com.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

CorelDRAW Graphics Suite

Choong Ng

Issue #84, April 2001

Corel's CorelDRAW Graphics Suite 9 is the first major commercial graphics package for Linux.

- Manufacturer: Corel Corporation
- URL: <http://linux.corel.com/products/draw/>
- Price: \$199 US
- Reviewer: Choong Ng

Corel's CorelDRAW Graphics Suite 9 is the first major commercial graphics package for Linux, combining both the object-oriented drawing capabilities of CorelDRAW and the bitmap photo editing capabilities of Corel PhotoPaint. If Graphics Suite 9 can provide end users and artists with good paint and draw capabilities, this will be a major step forward for Linux as a desktop operating system. Let's see how it measures up.

Installation

GS9 requires kernel 2.2 or higher and glibc 2.0 or better running on a 200MHz processor with 64MB of RAM and 255MB of free disk space. My test box substantially exceeded all of the requirements, but this actually ended up causing problems. Installation via the installation script went without incident, but due to incompatibilities with both XFree 4 and Red Hat 7.0 the FontTastic font system was not correctly installed. A short call to Corel's technical support, manual installation of the FontTastic software took care of the problem (I recommend calling Corel technical support for instructions specific to your system). Once the initial font problems were resolved, both Draw and Paint loaded without problems.

The Interface

For a user migrating from Windows, both CorelDRAW and PhotoPaint should have comfortably familiar interfaces. This is in part because Corel chose to use the WINE Windows compatibility layer (<http://www.winehq.com/>) to enable their applications to run on Linux with minimal changes. This approach has several implications from a technical standpoint, but from a user's perspective it simply means that you can expect Graphics Suite 9 for Linux to have the same features in the same places as Graphics Suite 9 for Windows.

While the user interface may be the same as it is on Windows, that by no means guarantees that the interface will be good. In fact, both CorelDRAW and PhotoPaint tended to have clunky interfaces and features that don't work quite the way one would expect. They also don't offer much in the way of key-stroke compatibility with their Adobe counterparts—that is the keyboard shortcuts in the two sets of applications are different and the Corel products are missing shortcuts to some often-used features. While their product may be targeted at a different market, Corel has attempted to duplicate all of the features and most of the interface of two applications (Adobe's Photoshop and Illustrator) that are targeted more at the graphics professional than the home user.

Features

There are both good and bad points to the effort to provide a duplicate feature set. On the good side is the relative ease of learning CorelDRAW and PhotoPaint once you know their Adobe counterparts. Most of the menu options are familiar and function much the same as they do in the Adobe apps, and PhotoPaint includes a complete set of functionally identical filters. CorelDRAW and PhotoPaint will even open .psd and .ai documents produced by their Adobe counterparts. On the downside is the fact that trying to replicate the working environment of someone else's software is rather difficult and, in this case, not very thoroughly done.

The biggest problem areas that I encountered were in PhotoPaint's relatively awkward handling of layers and layer transparency (the bread and butter of any image editor) and CorelDRAW's lack of keyboard shortcuts for many common functions.

Despite these deficiencies, Corel does do a fairly good job of adding features that a home user will appreciate, such as a helpful splash screen offering options to do common tasks such as creating a new document, opening the last one you worked on, viewing a tutorial or scanning an image. They also developed some of the best file load/save dialogs that I've seen in any Linux application; by default they open with your home directory as the root directory, but they still offer the option to look at other drives and everything

under the root directory via a drop-down menu. These and other features help make Graphics Suite 9 (and in particular the CorelDRAW portion) a worthwhile option for many people.

Technical Details and Other Notes

One of the most interesting things about Corel's Linux products is that they use WINE (the GPL'd Windows compatibility layer) to run what are otherwise essentially the same applications as Corel's Windows-based products. This isn't really a good or bad thing; there are several distinct advantages to this approach, but there are also major problems. I noted earlier that using WINE ensures that the Windows and Linux versions will function essentially identically, but WINE isn't perfect. I ran into problems with the main windows expanding beyond the bounds of the screen in both Corel applications. While easily fixable, a new Linux user would most likely not know the solution (holding down the Alt key and dragging the window to a position where a window border is visible, then resizing). Both packages also had problems rendering fonts on-screen, and I suspect that this too is at least partially a result of running through WINE. Though this is ultimately not a big deal in CorelDRAW—the visual artifacts do not appear in printed documents—it does present a problem in PhotoPaint where fonts get rasterized at screen resolution. These and other similar glitches make Graphics Suite 9 appear a little rough around the edges, but the glitches are for the most part offset by the features that Corel offers.

The Bottom Line

If you don't plan on doing a lot of graphics work—perhaps you are doing some lightweight graphics for your web site or building custom widgets for a GUI-based application—then Corel's Graphics Suite may be for you. On the other hand, if you plan on spending more than a few hours working with vector artwork or web production, you should consider looking for another solution such as The Gimp (<http://www.gimp.org/>). This is unfortunately one of those situations where retaining a copy of MacOS or Windows or a utility enabling you to run software for one of the former (such as Win4Lin, VMware, Plex86, WINE or Mac-on-Linux) may be necessary.

The Good/The Bad

Choong Ng (info@linuxjournal.com) is the product reviewer at *Linux Journal* and a teriyaki connoisseur.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

Firebox II

Glenn Stone

Issue #84, April 2001

This Seattle-based company makes a line of dedicated firewall appliances to serve anything from the SOHO market to a 5,000-user mega-office with up to 100 branches on Virtual Private Networks connected securely across the Internet.



- Manufacturer: Watchguard Technologies, Inc.
- E-mail: information@watchguard.com
- URL: <http://www.watchguard.com/>
- Price: \$4,990 US (Firebox II; options extra), \$449 US (base SOHO)
- Reviewer: Glenn Stone

Much ado has been made in the post-Melissa era about firewalling, particularly with the advent of low-cost, always-on high-speed Internet connections and associated security risks. Many of us here are familiar with the poor but educated man's approach to the problem: drop a Linux distro on a machine with a pair of NICs and configure appropriately. This is cheap, effective, reasonably secure, but also time-consuming and doesn't scale all that well. It's fine for Joe Hobbyist, but Mike the Network Manager wants something a little easier to handle.

Enter Watchguard. This Seattle-based company makes a line of dedicated firewall appliances to serve anything from the SOHO market to a 5,000-user mega-office with up to 100 branches on Virtual Private Networks connected securely across the Internet. Bundled with the bigger boxes is their LiveSecurity

System, a GUI-based manager tool that allows the network manager to easily configure the local firewall, called a Firebox, and optionally, VPN Manager, which allows management of multiple VPNs as well as any Fireboxes (including the SOHOs) that happen to be on the other end of the VPN link. As you might have guessed, the larger Fireboxes run Linux. (The SOHOs run VxWorks.) At \$4,990 US for a full-sized one, it's not cheap at all but, as I was to find out, it is easy.

Initial Impressions

The Firebox II arrived in a nice carry-handle box with all the cables needed to hook it up, including an RJ-45 crossover cable for setup, RJ-45 patch cords for the rest of the network and a serial cable for direct console setup with appropriate adapters for DB9 or DB25 connectors. The fire-engine red case also comes with a pair of metal flanges that, when you rearrange them appropriately, allow the Firebox to be mounted in a standard 19", 2U rack slot. The front panel has an array of LEDs for system status, traffic and load average; the back panel has three RJ-45 10/100Tx jacks, a pair of DB9 serial ports, a pair of Type II PCMCIA modem slots, power socket and switch. (I always liked the idea of the switch on the front, but that's just me.) Under the hood lies a custom single-board computer with a 200MHz Pentium MMX, 64MB of SDRAM, an 8MB Flash ROM and two noisy but effective fans. Instead of a CPU fan, there is a monster heat sink, and one of the case fans is aimed directly at it. I suppose this is so the unit can survive the loss of one of the fans; using a smaller fanned-heat sink would give the unit a single point of failure.



The manual is a nice 300-page spiral-bound volume about the size of a trade paperback. What lay inside, though, was the big surprise—Windows?!? *You need Microsoft Windows to run this software.*

A query to Watchguard's tech support web page netted me a pair of answering phone calls, verifying what the manual said. There is currently no way to set up the big Fireboxes without using Microsoft Windows. But, they said, we would be happy to have you visit our training facility and show you how things work.

On-site at Watchguard, we get down to business. We put the CD in the drive, the install ran, did its normal Windowsish thing (including a reboot) and now we're looking at the configurator for the Firebox. You can configure it for drop-

in mode (where it does transparent proxy like a bridge) or routed mode, which allows the trusted network to have private addresses, to which the Firebox will port-forward, if so desired. With all the requisite magic numbers in place, we pressed the button to upload and booted the Firebox. Hmm, uploading via TCP didn't work. Not to worry, you can also upload via the console port. Back up a click, reset for COM1 and up pops the progress box. Ahh, sweet success. A reboot under software control, and the Firebox is up and running. We verified that it would communicate over the trusted interface then used a pre-configured Firebox to address it over the external interface, set up an IPSec gateway and tunnel. The optional VPN Manager is really slick; you simply give it the remote address and configuration password of the remote Firebox, then drag one Firebox icon onto another, run through three clicks worth of configuration and the tunnel is configured; a quick reset (20-30 seconds) on each Firebox, and the tunnel is active. The VPN tunnels can be filtered in all the same ways you can filter regular IP traffic: by host, by port or both, on source, destination or both. The Firebox II client can also configure SOHO units remotely, so a network admin can manage his or her telecommuters as well as big branches.



I asked some pointed questions about the safety of uploading new configurations and got a neat insight into the Firebox's internal architecture. The Flash ROM is divided into several sections: the running configuration, the underlying Linux system, a backup area where these can be saved (and recovered) during upgrades and a "system" area, which is the moral equivalent of a rescue partition. You tell the configurator you want to restore the factory default configuration, then reboot the system with the console port connected to your serial port. The configurator detects the boot prompt and tells the system to boot from the system area, at which point you reconfigure the machine from scratch. You can also boot the box with a PCMCIA modem in one of its slots and (re-)configure the machine from remote dialup. This makes for easy physical deployment of a large VPN; all the person at the remote site has to do is insert various sets of tab "A" into slot "B" (and the cables are all color-coded) and turn it on, and the network admin sitting in the home office in Sioux City can take it from there.

Saving Grace

This is all well and good for larger companies with heterogenous networks, but what about us Linux-only types with a relatively tiny network and budget? (After all, the Firebox II is rated at 500 users and costs five grand.) Not to worry. Remember those SOHOs? Well, they don't run Linux, but they do **grok** it...as a matter of fact, any SSL-capable browser can configure a SOHO. The configuration screens are all straight, low-graphics HTML, so, while it doesn't look as fancy as the Windows client, you could even talk to it with Lynx patched for OpenSSL. They're also a lot more affordable, with an MSRP of \$449 US for a ten-user version. It's about the size and shape of an 8-port hub with a built-in 4+1-port hub on the back and a few status LEDs in the front.

All the Fireboxes do NAT, logging, DHCP both client and server, logging to a remote host, remote setup of one form or another and, if you're stuck with it, PPP over Ethernet. VPN is standard on the big boxes, and a \$100 US option on the SOHOs. (Interestingly enough, the VPNs use IPsec and a few other protocols, but not Microsoft's PPTP. Why? That's right. Microsoft wasn't forthcoming with the standards.) The big boxes also perform scan and spoofing detection. All of them come with a year's subscription to LiveSecurity, a "push" service that delivers security updates, both human readable and in software form, via e-mail or directly to the configuration host. Network managers can then upload the software at their convenience. Your subscription also gets you access to the tech support web pages, which include both a knowledge base and a trouble-ticket submission and tracking interface. The license key is also your password to get tech support on the phone. (Don't lose that card!)

The Future of the Firebox

Watchguard said they were working on ways to minimize the number of times you had to reset a Firebox after a configuration change; this makes sense since you can do almost anything except change kernels to a standard Linux machine without resorting to a reboot. They also hinted that there might be Linux in the SOHO's future. For the foreseeable future, however, the big Fireboxes will require a Windows host to configure. Watchguard's primary focus is to make it as easy as possible to deploy a network on a truly grand scale and manage it with a minimum of fuss.

They've done that to a certain extent already. As the old saw goes: good, cheap, fast—choose any two. This isn't cheap, but I think with the dual solution of the Firebox IIs for large, heterogenous networks and the SOHOs for small or Linux-only implementations, a network admin can make a case for these drop-in machines in terms of saved people-time, both up-front and on an ongoing basis. It's not a perfect solution, particularly not for the purists among us, but I

think it's a step in the right direction. Not bad for a company "only" five years old.

The Good/The Bad



Glenn Stone (taliesin@speakeasy.org) is a Red Hat certified engineer who thinks AOL CDs make great target practice. He is a participant in the Seattle Linux User's Group mailing list.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Letters

Various

Issue #84, April 2001

Readers sound off.

Out of the Scope

Regarding “A Survey of Embedded Linux Packages” (*LJ* February 2001), thank you, thank you and thank you. I couldn't see the forest because all them damn trees were in the way. Your article really helped shed light on what the names are, who the players are and what the scope of the paradigm is.

Living in Northwestern Pennsylvania, we are kind of out of the Silicon Valley mail loop. I've been on Victor Yodaiken's RTL mailing list for two years and still didn't understand the landscape. Your article really helped.

Thanks for helping us stay current.

—Raymond C. Minich

Best of Tech Error

There is an error in “The Best of Technical Support” in *LJ* February 2001. “ide-floppy” is not the appropriate module for ATAPI class ZIP drives. ide-floppy works for the original non-ATAPI IDE ZIP drives. The appropriate module is “ide-scsi”. This information may be found in the ZIP drive HOWTO.

This is a very serious difference. From firsthand experience, I know that using ide-floppy on an ATAPI class ZIP drive will seem to work but will invariably cause filesystem damage on the ZIP disk (unless one does read-only operations with it).

Sincerely yours,

—Michael O'Brien

Less-than-Secure?

One of the examples in Mick Bauer's "101 Uses of SSH, Part II" article (*LJ* February 2001) gives a misleading impression of security. Listing 6 shows how FTP can be tunneled over ssh port-forwarding. If this were any other protocol (say POP3 or IMAP), things would be fine.

But FTP actually uses *two* connections, not just one. The primary connection (port 21) is used as a control channel, issuing commands to the server. The secondary channel is set up each time there is data to be transferred on a different and random port, if using passive mode. If you're not using passive mode, the situation gets worse. The server tries to make a connection back to you from port 20. Chances are this will be blocked stone dead by any firewall nearby.

Anyway, the **ssh** command will only forward the control channel and not the data channel. That is enough to protect your password but not the data that passes between the servers. This is a misleading state of affairs because the connection may well work, even though most of it goes unencrypted!

As far as I know, you cannot use FTP over SSH without implementing special "FTP watcher" routines inside ssh. This is not impossible; practically every NAT device on the market does the equivalent. An alternative might be to use the **sftp** command from ssh2. However, I'm not sure whether this is included in OpenSSH yet.

—Dominic Mitchell

You're absolutely correct. While I do think there's some value in encrypting the control channel even if the data channel is not, that was a poor example, especially since I didn't point out that the data channel does in fact go in clear text. But it does work, even without an FTP-watcher: I've been using FTP over SSH in the manner described in my article for a couple of years now. SCP is still the most secure way to transfer files with OpenSSH. SFTP is cool, but only partially supported in OpenSSH (client- or daemon-only, I can't remember which). But you can use SSH Communications' "official" SSH v.2 for free if you run it on Linux, xBSD, etc., thanks to a new "Free Use for Open Source OSes" clause in its license.

—Mick Bauermick@visi.com

Give Us More

Out of all the articles I've read in *LJ* over the years, few have had as much impact on me as the one on **vim** ("That's Vimprovement! A BetterVi", *LJ* February

2001). I often spend more time in vim than I do sleeping, and thanks to the article, I now have a few more tricks to use. Some of the most glaring features I was not aware of were the third mode (called visual), the command mode **grep** and **split**. I learned vanilla **vi** on Solaris, and there are a lot of features in vim I have obviously not picked up yet. That being the case, have you thought about a monthly feature on editors? Since I imagine a great deal of your readers spend a lot of time in editors such as **emacs**, vim, **pico** and the like, perhaps a regular feature is in order.

—Robert Lazarski

Sober Prediction

Regarding your 2000 Editors' Choice Award for open-source databases (see *LJ* December 2000), I disagreed with your pick for the year's best OS database. As per your suggestion, I had a stiff Tom Collins, followed by some White Russians *and* a few shooters. I became light-headed, then dizzy and finally passed out before reconciling myself to your pick. Having sobered up, I now take to my keyboard for a more formal protest; I suggest your readers contact NuSphere and evaluate their version of the MySQL database. While I realize we'll just have to live with your pick for 2000, I predict that the Editors' Choice for OS databases in 2001 will be NuSphere MySQL—hands down and bottoms up.

—Bill McGowan

We met with NuSphere at LinuxWorld this past February, keep an eye out for a formal product review in an upcoming issue.

—Editor

Oops?

Okay, was this on purpose or not? (If it was on purpose, then it was extremely funny; otherwise it was also extremely funny but in a completely different way.) I think the quote, "Beware of bugs in the above code; I have only proved it correct, not tried it," should be attributed to the great Donald Knuth, not Donald Knut (*LJ* February 2001, page 14).

—Keir Davis

Martha Stewart Linux

I read your editorial (*LJ* February 2001, page 93) in which you mentioned that *LJ* is two-thirds the size of *Martha Stewart Living* with one-tenth the staff. You also mentioned that Martha doesn't have to lay out Perl code.

I wanted to let you know that we definitely do lay out Perl code here at Martha Stewart Living Omnimedia. Don't forget, we have an e-commerce site (<http://www.marthastewart.com/>) that does several million dollars' worth of business each year. Also, our magazine staff puts out three regular magazines (*Living*, *Babies* and *Weddings*), as well as many special editions.

Presently, our web site is running on NT servers, but we are in the midst of migrating our site to Sun Solaris. And, you might be interested to know that the server that streams media to the site is running Red Hat Linux 7.0! Many of us here at Martha Stewart are open-source fans with a particular interest in Linux. I personally have been reading *Linux Journal* since issue number one. As a developer, I am constantly looking for ways in which we can use Linux at work. Check the header of this message, and you will see it is coming to you on a Linux machine with Pine.

I wasn't offended by your article in the least, I just wanted to let you know that even companies like Martha Stewart Living are embracing Linux.

Best regards,

—Rick Noelle
Internet Applications Developer
Martha Stewart Living Omnimedia

Need a Calendar

In reading the February 2001 issue of *Linux Journal*, I noticed the article "Linux and the New Internet Computer". There is a screen shot of the NIC with some apps running on it. Strange as it may sound, I'm very interested in the calendar program running on that screen.

I've been using **xcalendar** for years now, and I love its simplicity. However, it seems that it's not being maintained. The calendar running on that screen shot looks similar but different. I'd like to find out if it is a *different* minimalist calendar program.

Thank you.

—Michael George

The X "calendar" client you see on the desktop in the screen shot, you're absolutely correct, it is "minimalist"! Here it is:

```
cal | xmessage -file - -title "Calendar"
```

a better version is with the **gcal** command (which highlights the current day):


```
gcal | xmessage -file - -title "Calendar"
```

If you use the **tcal** command, tomorrow's day will be highlighted. Of course, if you have a big display (1600 <\#215> 1200), you can put up a year's calendar with:

```
cal -y | xmessage -file - -title "Happy New Year"
```

I know that this is a brain-dead calendar—if you want something with a bit more intelligence that can launch calendars, reminders and check your mail, look at the **rclock** client...its man page gives all sorts of interesting applications (such as changing your desktop's hues according to the time of day—brightest at noon, fading to darkness toward evening, etc.).

—Billy Ball www.tux.org/~bball

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

UpFRONT

Various

Issue #84, April 2001

Stop the Presses, *LJ* Index and more.

LJ INDEX—March 2001

1. Power consumption per day of an IBM z900 (formerly S/390) mainframe: \$32 US
2. Starting price of an IBM z900: \$750,000 US
3. Number of Linux instances that have been run on one z900: 41,000
4. Percentage of companies that have deployed, or intend to deploy, at least one Linux system: 68
5. Number of US information-appliance unit sales in 2000: 7,440,000
6. Projected US information-appliance unit sales in 2005: 51,800,000
7. Compound annual growth rate of US information appliance sales: 47.4%
8. Worldwide information-appliance unit sales in 2000: 29,000,000
9. Projected worldwide information-appliance unit sales in 2005: 305,000,000
10. Compound annual growth rate (CAGR) of worldwide information-appliance sales: 59.8%
11. Number of BSD-based Apple OS X betas downloaded or shipped by January 2001: 100,000
12. Number of people submitting input to the OS X beta process: 70,000
13. Volume of résumés posted per day on Monster.com: 38,000
14. Monster.com's total job database: 12,000,000
15. Pounds of ground beef that might be contaminated by one diseased animal: 32,000
16. Chances that an American will have a fast-food meal today: 1 in 4
17. Number of class action lawsuits filed against VA Linux in January 2001: 5

18. Number of those lawsuits that erroneously referred the company as "Linux": 4
19. Number of similar lawsuits filed by one of those firms, Milberg Weiss, in one decade: 200
20. Estimated cost in millions of dollars of a complete Linux solution on one IBM mainframe: 7
21. Estimated cost in millions of dollars of the same solution on equivalent Sun hardware: 55

Sources:

- 1-2: CRN
- 3: LinuxPlanet
- 4: CIO
- 5-10: eTforecasts
- 11-12: Apple
- 13-14: ZDNet
- 15-16: Eric Schlosser, author of *Fast Food Nation*
- 17-18: *Linux Weekly News* (www.lwn.net)
- 19: law-phoenix.com
- 20-21: *Infoworld*

The Kernel Speaks

Which computer architecture is best? We got the Linux kernel source, version 2.4.0, and counted the instances of George Carlin's "seven words you can never say on television", plus "crap", "damn" and "sucks" in each subdirectory of /usr/src/linux/arch.

The results:

alpha: 1arm: 0i386: 7ia64: 0m68k: 3mips: 22mips64: 5parisc: 4ppc: 3s390: 0sh:
1sparc: 19sparc64: 13

So, it's clear that ARM, IA-64 and System/390 are, from the kernel developer's point of view, the best computer architectures, and as for those piece-of-[expletive deleted] MIPS and Sparc boxes, well, the less we [expletive deleted] say about them the better.

—Don Marti

Linux Bytes Other Markets: Bay Area Rapid Transit (BART): Under Control with Linux

by Gary A. Messenbrink and Frank Ruffa

In the Operations Control Center (OCC), 20 feet underground in the heart of Oakland, California, the Bay Area Rapid Transit (BART) system is managed by a group of dedicated individuals tasked with moving residents quickly and safely from one end of the San Francisco Bay area to the other.

Along the sides of the room are 30 projection TV systems that display the Train Control map for the system and the Traction Power Electrification Display. The TV projection systems are driven off of NCD X terminals that connect to a Tandem S4000 that maintains the state of the system. The controllers use Sun workstations to query and manage the system, which in turns communicates to the Tandem.

The extension of the BART system to the San Francisco Airport required an upgrade to the Electrification Display Systems of the OCC to be available by July of 2000. In planning to support this upgrade, we decided to improve some of the human factors relative to managing the system too.

The system was originally designed to use multiple high-resolution (for the time) Tektronix displays. As a result, experienced controllers could quickly glance at these displays and immediately grasp the state of the system. In moving from the Tektronix displays to Sun workstations, the concept of overlapping windows was introduced because of limited screen real estate. Although functionality was increased, the overlapping displays were not popular with the controllers as they lost the ability to understand the system at a glance.

Sony has recently released 24-inch monitors with larger screens. With the wider 16:9 aspect ratio, these monitors were ideal for the type of landscape display that BART needed to eliminate overlapping windows. Unfortunately, the Sun Solaris 2.4 operating system didn't provide support for these new monitors as the resolutions required, forcing us to investigate other options.

We quickly found that the open-source nature of Linux provided the solution. Supporting the new monitors required nothing more than some simple changes to the configuration file used by XFree86. With this knowledge, we selected an economical off-the-shelf PC system driving the Sony 24-inch monitor running Linux as our new OCC workstation.

We selected Motif as our user-interface toolkit, instead of using Qt or GTK+, based its proven rock-solid stability in mission-critical applications. Then we started looking for software that would speed the development of user

interfaces and deliver an improved graphical display of the system. We found solutions to both of these needs with LOOX++ from LOOX Software (<http://www.loox.com/>) and Builder Xcessory PRO (a 1999 *Linux Journal's* Editors' Choice award) from ICS (<http://www.ics.com/>). LOOX++ simplified the visualization of our Electrification System by providing us with the ability to easily create a graphical display for our model. We used Builder Xcessory PRO to quickly build and tailor the graphical user interface for the application.

The development of the new electrification software started in December of 1999 and took approximately three months, meeting the required date to support the San Francisco Airport Extension. The Suns that once ran the controller workstations have been retired, and Linux is now the workstation used by all controllers in the OCC.

Given the potential for human tragedy resulting from either a hardware or software failure, our selection of Linux was initially a politically charged issue. However, we have demonstrated considerable success with the new Electrification System and have recently been given the approval to initiate the second phase and convert the Train Control System to Linux, too.

The use of commodity PC hardware, not having to purchase software licenses for the operating environment and the rapid development tools resulted in a cost savings of approximately 15-20% of the project budget. The performance record of the Linux environment has been flawless despite the 24/7 active operation.

Gary A. Messenbrink is a principal computer systems engineer and long-term BART employee. Frank Ruffa is a manager of IT development at BART.

Penguins on the Concept of Teamwork

In *The Penguin Post* (yes, it exists, www.penguin-place.com) we recently found two reports of great interest and no importance suggesting that penguins are unusually sporting birds.

Item #1: Bird Bowling: The British Air Force in the Falkland Islands discovered a way to “bowl” for penguins by flying over large groups of the groundbound birds. See, penguins are observant creatures that are unusually interested in airplanes (flight envy perhaps?). Herds of up to ten thousand penguins will, in unison, all point their beaks at a plane flying overhead, tracking it accurately as it passes by. Sooo...if the pilots fly over the penguins at just the right angle, the little animals all fall over backwards—again in unison—just as the plane passes overhead.

Item #2: Birdball: The penguins hanging around the Woods Hole Oceanographic Institution in Antarctica have been observed to take interest in games of football played by humans on flat fields of snow and ice.

One day the scientists came out to discover that the penguins had taken the field. They would line up in two rough groups, and then start squawking and running around bumping into each other. After a bit of this, they would pick themselves up and start the process all over again. They hadn't quite got the idea that a ball was important to the process, but they kept at it for some time. The sight was so ridiculous that everyone was rolling on the ice laughing.

Stop the Presses: Kylix Clix with CLX

Borland, the development tools company that recently returned to its original name (shedding the confusing "Inprise"), has finally come out with Kylix, its long-awaited rapid application development (RAD) Integrated Development Environment (IDE) for Linux—and for cross-platform development as well.

Developed entirely for Linux as a native Linux development toolset, Kylix is also very familiar to users of Delphi, Borland's popular Windows RAD toolset. Kylix not only allows developers to use nearly identical tools and skills but to move Delphi-developed Windows applications over to Linux. Borland claims there are over two million Delphi and C++ Builder users, millions of applications, thousands of vertical and horizontal components, and over five hundred registered third-party tool and component vendors that will easily leverage from the Windows world to Linux using Kylix.

At the core of Kylix is the Component Library for Cross Platform Development, or CLX, pronounced "clicks". This is an open-source application infrastructure that supports both the commercial version of Kylix and the core component architecture, which Borland is releasing under the General Public License (GPL). These are expressed in three product editions: Open Edition, Desktop Developer and Server Developer. All three are available as shrink-wrapped products with the Open Edition available for free download as well. All CLX class libraries, however, are open source and GPL'd.

Like other commercial Linux vendors, Borland is attempting to support both commercial application development and ubiquitous Linux-based infrastructure. There is plenty of demand for both. This was manifest before the Kylix project began, when Borland conducted an extensive survey (partly through *Linux Journal*) in 1999. The survey had more than 24,000 responses that included both Linux and Windows developers. RAD development tools were at the top of the wish list for both groups. The top types of planned

applications were application/utility development and client/server database development.

Ted Shelton characterizes Kylix's potential impact this way:

Imagine what would happen if Microsoft open sourced .Net, the crown jewels of their new strategy to dominate the emerging market for “web services”. Imagine how thousands of developers would be able to take .Net and not just build applications with it, but truly extend it—move it to new operating systems, add functionality, specialize key interfaces for new devices, environments and vertical applications. Imagine how fast the world would change. We're ready to provide the drills, hammers and circular saws for developers to build an entirely new application infrastructure, one that will be industrial strength, open source and Linux-ready.

After the press conference announcing Kylix, it appeared to me that the deepest significance of Kylix may be in its cross-platform class libraries—in CLX. During the conference, Borland CEO and President Dale Fuller compared CLX to .Net, calling CLX “.Now”. I asked Ted Shelton if this positioned CLX as a cross-platform framework that competed not only with .Net (which is not really cross-platform but cross-language within the Windows platform), but with Java as well. He replied, “When you develop, you essentially choose a set of class libraries. Java offers one. Microsoft offers another with .Net. CLX is now the third. But it's the only one that is both open source and cross platform. I expect that after the Open Source community gets into it, the CLX libraries will outnumber Java's.”

They Said It

We need to start teaching programming and hacking in first grade or, better, earlier.

—Bob Frankston

First we thought the PC was a calculator. Then we found out how to turn numbers into letters with ASCII, and we thought it was a typewriter. Then we discovered graphics, and we thought it was a television. With the World Wide Web, we've realized it's a brochure.

—Douglas Adams

Wade's Maxim: No one ever made money by typing.

—Wade Hennesey

The moon is covered with the results of astronomical odds.

—onyxruby on Kuro5hin.org

Linux is like a wigwam: no windows, no gates, Apache inside.

—Albert Arendsen

A thinking computer...you mean, like a swimming ship?

—Albert Arendsen

We cannot trust some people who are nonconformists. We will make conformists out of them in a hurry...the organization cannot trust the individual; the individual must trust the organization.

—Ray Kroc, founder of McDonald's

We reject kings, presidents and voting. We believe in rough consensus and running code.

—David Clark

Win on Lin on Thin

The New Internet Computer Company (better known by its acronym, NIC, and perhaps best known as the creation of Oracle CEO Larry Ellison, who owns it personally) has teamed up with Menta Software to offer the equally inexpensive and ironic combination of Windows apps running across the Net on a \$199 US Linux “thin” client. The NIC, which was shown running Menta's “thin server” WinToNet at Linux World Expo in New York, is designed for schools and other “price sensitive” networked environments.

We asked Gina Smith, NIC's CEO (and former high-profile journalist) to give us the skinny on adding value to extra-thin devices. “It is really cool”, she said. “Basically, our system is a super-affordable hard disk-free Linux client. We have 56K modem and Ethernet connections built in. Using Menta's Java app, we can run Windows apps from a server over the Internet.” Adds Menta's Bruce Fryer, “Why put fat apps on a thin client? When people see WinToNet running on the NIC, they are blown away.”

What are their chances? Consider these two facts: 1) their sole stockholder is worth a few dozen billion dollars—give or take a few billion a day; and 2) their home page features a prominent link that reads “GNU General Public License”.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Lunacy Floats

Doc Searls

Issue #84, April 2001

Doc announces the October 2001 Geek Cruise.

What's 750 feet long, weighs more than 50,000 tons, floats on blue Caribbean water and has an IQ upwards of 190,000 that will only get higher by hanging out with itself for a week?

Try the Linux Lunacy 2001 cruise, which sails October 21-27. Copresented by GeekCruises.com and *Linux Journal*, Linux Lunacy will cruise between ports in the Bahamas, the Virgin Islands and Puerto Rico in a Holland America ship featuring ten passenger decks and educational sessions on programming, languages, tools and issues ranging across free and open-source development cultures, privacy, hacker politics, ham radio and troubleshooting IT projects—among many other things.

Leading these sessions is an all-star cast of Linux Luminaries (and leading lunatics):

- Richard Stallman—the world's leading authority on Free Software, a distinction he earned as founder of the GNU Project and author of its leading works, including GNU C compiler, Emacs, GDB and other tools that went into building what we call GNU/Linux, along with much of the common computing infrastructure we all enjoy today. His work has been recognized by Grace Hopper and MacArthur Foundation awards, among many other honors. Richard will teach classes on Emacs Lisp Programming and Emacs editing.
- Eric S. Raymond—self-described hacker and anthropologist who studied the Hacker tribe for a decade and a half before achieving notoriety for his speaking and writing, which have together made him the prime figure in the Open Source movement. His best-known work is *The Cathedral and the Bazaar*, the influential paper that became an influential O'Reilly book

by the same name. That book combined a series of papers that serve as the canon for the Open Source movement. A cofounder of the Open Source Initiative, Eric's name is also on the spine of *The Hacker's Dictionary*, now in its third edition. Eric will speak on "Twenty Years among the Hackers".

- Guido van Rossum—the creator of Python, an open-source scripting language that is exploding in popularity. Guido created Python in the early 1990s at CWI in Amsterdam and still leads development of the language. He moved to the US in 1995 and now works for Digital Creations, makers of Zope, the open-source web application platform that is written in Python. At Digital Creations he is director of PythonLabs, the core Python development group. Guido will teach a half-day introduction to Python.
- Bruce Perens is a major GNU/Linux developer, cofounder of the Linux Capital Group, cofounder of the Open Source Initiative, and founder of the Linux Standard Base among many other distinctions. Bruce will teach classes on Debian development and speak on subjects ranging from "Hacking Politics" to the "Economy with Free Software" to venture capital and ham radio.
- Jon "Maddog" Hall—the executive director of Linux International, the leading nonprofit organization promoting the use of Linux. His work with LI has been funded since 1999 by VA Linux Systems. As a familiar figure in the Linux community, Maddog is perhaps second only to Linus Torvalds himself (in fact he is godfather to Linus' children). As a Linux veteran, Maddog has a history that goes back through Compaq to Digital Equipment Corporation to IBM to Bell Laboratories' UNIX group. Among many other achievements, Maddog was directly responsible for the port of Linux to the Alpha processor. Maddog will speak on "Linux in Education" and "Task-Oriented: Where Linux Falls Down".
- Art Tyde is cofounder and CEO of LinuxCare, the leading Linux service and support provider. He was president of the Bay Area Linux Users Group (BALUG) when he founded LinuxCare with fellow BALUGer Dave Sifry (and he's still the president there). Art is a corporate information systems veteran and IT disaster planning and recovery expert who will be giving a half-day "Introduction to SAMBA".
- Dave Sifry is cofounder, CTO and VP of Engineering for LinuxCare, the company he founded with Art Tyde when both were leading Linux lunatics at BALUG. An authority on open-source development, Dave is still vice president of BALUG and serves on the board of Linux International. Dave will give a keynote speech on "The Future of Open Source".
- Barbee Davis is one of the leading authorities on project management, owner of Davis Consulting and ExecuTrain and coauthor of Macmillan's *How to Learn Microsoft Project 2000 in 24 Hours*. A certified project management professional (PMP) and a veteran of countless major IT

projects, Barbee has had an impressive career working with corporations ranging in size from IBM on down to the five new enterprises she nurtured from concept to completion, winning numerous awards along the way. Barbee will give a full-day session on “Why 90% of All IT Projects Fail—and What To Do about It?”.

- Randal L. Schwartz—one of the world's leading authorities on Perl. He coauthored the must-have standard texts on the subject: *Programming Perl*, *Learning Perl*, *Learning Perl for Win32 Systems* and *Effective Perl Programming*. He is also a columnist for *WebTechniques* and *UnixReview* magazines. His company is Stonehenge Consulting Services, Inc., which he has owned and operated since 1985. Randal will teach a full day of “Learning Perl”.
- Simson L. Garfinkel is chief scientist at Broadband2Wireless and chief technology officer at Sandstorm Enterprises, a computer security company that develops offensive information warfare tools used to probe the security of computer systems and test defenses. Simson is also a veteran journalist whose byline has appeared in *The Boston Globe*, *The San Jose Mercury News*, *The Christian Science Monitor*, *Technology Review Magazine*, *ComputerWorld*, *Forbes*, *The New York Times*, *Omni*, *Discover* and *WIRED*, where he was a founding contributor. He is also the author or coauthor of nine books, the most recent of which is *Database Nation: The Death of Privacy in the 21st Century*. Simson will speak on privacy and data encryption.
- Michael K. Johnson—has been working with Linux since Linux kernel version 0.02 was released. He has contributed to the kernel, utilities, toolkits, applications and documentation and is the coauthor of *Linux Application Development*, published by Addison-Wesley. He helped build Red Hat Linux as a developer for several years and is now Red Hat's manager of kernel engineering. He is also a private pilot, voracious reader and small-time tool aficionado.
- Reuven M. Lerner—has been writing the popular “At the Forge” column for *Linux Journal* for over five years and published the first newspaper on the Web just after graduating from MIT. Named a “Web pioneer” by publisher O'Reilly and Associates, Reuven now runs a consulting company specializing in the creation of database-backed web sites that use a wide variety of open-source operating systems, utilities, databases and programming languages. Reuven lives halfway between Jerusalem and Tel-Aviv, in Modi'in, known as Israel's “city of the future”. Reuven will conduct a half-day “Application Server Shoot-Out”.
- Steve Roberts—was driven by “a combination of chronic restlessness and midwestern torpor” into a “technomadic life”. That was when, to the horror of friends and family, he sold his suburban house, moved to a

recumbent bicycle and began a 17,000-mile bicycle trip around the US—stopping now and again to write books about the adventure or rebuild the substrate. The third version of the bike, “Behemoth”, sported 72 watts of solar panels, a network of on-board computers with handlebar chord keyboard and head mouse, heads-up display, satellite Internet link, 105 speeds, ham shack and other goodies. By 1993, he was at work on the bike's aquatic successor, the Microship, which is based on a pair of canoe-scale amphibian pedal/solar/sail micro-trimarans with on-board Linux servers, live telemetry to a public server, video production tools, a complete suite of communications resources, deployable landing gear and 480 watts of peak-power tracked solar panels per boatlet, among other things. Steve will give a “Canoe's Not UNIX” talk and participate in various BOF (Birds of a Feather) sessions.

- Michael “Monty” Widenius (like Linus, a Swedish-speaking Finn)--has been working with databases since 1978. He has been employed by and is part owner of TCX since 1981 and has written all of UNIREG (the predecessor to MySQL) and most of the MySQL server. He has studied physics at the Helsinki University of Technology but is mostly self-taught in BASIC, Assembler, C, LISP, Perl, SQL and C++; he is always willing to look at new languages when he gets the time. Monty will be speaking on “History and Introduction to MySQL”.
- Doc Searls (yours truly)--senior editor of *Linux Journal* (being the oldest on the editorial staff) and of its new sister publication *Embedded Linux Journal*. He's also coauthor of *The Cluetrain Manifesto*, a *New York Times*, *Wall Street Journal* and *Business Week* best-seller. His byline has also appeared in *OMNI*, *PC Magazine*, *The Globe and Mail*, *Upside* and *WIRED*, among many other publications. At *Linux Journal* he obsesses out loud on issues of business, infrastructure-building and all the ways commerce and geekery get along without ever knowing it. He'll be speaking on one or more of those subjects.

And then there are the rest of the lunatics who sign up.

The number of technical sessions totals more than 30, plus there will be countless opportunities just to hang out, talk and have both geeky and other kinds of fun.

The conference fee is \$750 but goes up to \$875 after July 15, 2001 and includes all courses, course materials and the Bon Voyage and Wizards Cocktail Parties. Spouses and guests are welcome to join in the entertainment.

Doc Searls is senior editor of *Linux Journal* and a coauthor of *The Cluetrain Manifesto*.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

Best of Technical Support

Various

Issue #84, April 2001

Our experts answer your technical questions.

Programming with Shared Memory Segments

I have some processes that create **shm** segments and others that link to those segments. The problem is that some processes create shm segments with a key, but the addresses of the attachment are the same as that of previous shm segments created by other processes with different keys. —Christian Grunfeld, cgrunfeld@uolmail.com.ar

There is a good Linux shared memory tutorial at www.acm.uiuc.edu/lug/presentations/shm/shm.html. —Felipe E. Barousse Boué, fbarousse@piensa.com

Nice Keyboard, but Where Are the Accent Marks?

How do I enter a foreign/high-ISO8859 character, for instance ê, from inside an X Windows System? At a console screen, I can do Alt+237 and get that character. Various documents talk about the “Compose” key, but I'm unable to get results by following any of those instructions. —Kevin Goess, knet@goess.org

I was able, by building an **.Xmodmap** (notice leading dot) file, to remap all the keys in order to accept Spanish, English, Portuguese and French characters by using dead-keys, CTRL, ALT and ALTGR sequences. I will put my `/usr/lib/X11/xinit/.Xmodmap` on-line at <http://www.piensa.com/xmodmapfile/>. This file is “run” whenever X is initialized. You can run it manually with the **xmodmap** command. —Felipe E. Barousse Boué, fbarousse@piensa.com

A GUI utility to pick special characters is the Gnome Character Map, available at <http://www.gnome.org/>. —Paul Christensen, pchristensen@penguincomputing.com

Login Won't

I'm running a small home network with Linux/Win95/WinNT all coexisting. Recently, the Linux machine stopped allowing Telnet sessions or direct (text) logins on the console. I'm receiving this message: **/sbin/login: /lib/libc.so.6: version `GLIBC_2.1.3' not found** (required by /sbin/login)

The file /lib/libc.so.6 is linked to /lib/libc-2.1.1. Can you tell me what I am missing? —Larry Busse, ljb@one.net

It looks like you upgraded /bin/login (probably through the **util-linux** RPM), and you now have a dependency problem since your binary was built for glibc 2.1.3, and you have glibc 2.1.1. I'm not quite sure how the RPM was installed, considering that it should have complained due to library incompatibility. One possibility you should be looking at is that someone accessed your system without your knowledge and installed a modified /bin/login to compromise your system. If that's not the case, re-install the util-linux package that came with your RH 6.0 distribution (**rpm -U --force**) and you should be okay. —Marc Merlin, marc_bts@valinux.com

CD-ROM Mounts...but Where?

I'm trying to install Sun StarOffice 5.1 on my machine. I mount the CD-ROM (mount /mnt/cdrom), but when I go to the cd-rom directory (/cd) to try to see the files, nothing is there. —Luis Embalo, zulobaby@hotmail.com

Make sure you're looking in the right directory. If mount /mnt/cdrom doesn't give you any errors you should be looking in /mnt/cdrom for the contents of the CD-ROM. One thing you can try is using the mount command with no arguments. On my system it returns this line:

```
/dev/hdc on /mnt/cdrom type iso9660 (ro)
```

This tells me that my CD-ROM drive (hdc) is mounted at /mnt/cdrom. If you would like to mount your CD at /cd, try:

```
mount /dev/cdrom /cd
```

--Paul Christensen, pchristensen@penguincomputing.com

Full-Duplex or Not?

In an HP-UX system, to find out a duplex status of an Ethernet card, one simply issues a command: **lanadmin -x lan0**. This will report if a card is in full-duplex 100MB mode or not. I've been trying to find out for a long time what would be

an equivalent of something similar in Linux but cannot find it. —Boleslaw Mynarski, bman@bolek.com

Sure, you have two of them: **mii-tool** and **mii-diag**. One place you get those from is: <ftp://ftp.valinux.com/pub/support/flory/mii-tool/>. —Marc Merlin, marc_bts@valinux.com

Setting up Apache jserv

I'm trying to install Apache jserv on Linux Mandrake 7.2. The installation process installs Apache for me. The path given in books and in Apache jserv documentation says the default directory is /usr/local/apache or /usr/local/src, but I don't have either directory.

I have also tried an example from the book *Java Programming on Linux*:

```
./configure --with-apache-install=/usr  
--with-jsdk=/usr/local/Java/JSDK2.0
```

It returns the error **./configure no such file or directory**. Any help would be greatly appreciated. —Christopher Nallo, cnallo1@home.com

/usr/local/apache is where Apache is installed if you compile and install the application. Otherwise, it was probably installed as an RPM installation file. If you do:

```
rpm -qa | grep apache
```

you will find how many Apache components you do have installed, including which versions. If you do:

```
rpm -ql <apache-package>
```

where <apache-package> is one of the output lines of the first command, you will get the location where each file of that package was installed. Regarding ./configure you should run configure files from the directory in which they are located. You are sitting somewhere else in your directory tree, not where the configure file of the package you want to install is located. —Felipe E. Barousse Boué, fbarousse@piensa.com

Install Problems on NEC Laptop

I have been trying to install Caldera 2.3 on my laptop, an NEC Versa 4230, and have run into major problems! First, I can't get the X server up and running at any decent resolution. Second, even though I get the basic system installed and

supposedly running, it will not boot; it just locks—Ed Money,
edmoney@turbosport.com

If you have an old X server it will only support your chip in VGA mode, which is bad. Upgrade to XFree86 3.3.6 or better. —Marc Merlin, marc_bts@valinux.com

Check out <http://www.cs.utexas.edu/users/kharker/linux-laptop/> and search for your laptop model. You'll find lots of useful notes and links from people who have made it work. —Paul Christensen, pchristensen@penguincomputing.com

X on Intel 810

I have an Intel 810 chip set with an onboard sound and graphics card. My problem is that I cannot configure X to work properly. I want to know if I can get device drivers for the above-mentioned chip set? —Ashutosh,
a_m_pandey@hotmail.com

You need a fairly recent version of XFree86; 3.3.6 should work according to <http://www.xfree86.org/3.3.6/RELNOTES4.html#4/>. You can also upgrade to XFree86 4.0.2, which supports your chip set, too, <http://www.xfree86.org/4.0.2/RELNOTES3.html#8/>. —Marc Merlin, marc_bts@valinux.com

Blanking in VTs

After a period of inactivity, my terminal blanks. For various reasons, this is less than desirable. How can I extend the timeout or disable the blanking? I am not in X, but at the 80 x 25 screen. —Garth, garthb@digitalwave.com

The command you're looking for is **setterm**. The format is simple:

```
setterm -blank 10
```

This will give you 10 minutes of terminal inactivity before the screen goes blank; 0 (off) to 60 is valid. If you add

```
setterm -powerseve powerdown
```

your monitor will actually go into sleep mode to conserve energy. —Paul Christensen, pchristensen@penguincomputing.com

Using a Parallel-Port Zip Drive

I want to be able to access my Iomega ZIP250 parallel drive. —Mark Adams,
mradams@accusys.com

This drive uses a module called **imm**. At the command line, while logged in as root, type **modprobe imm**. This will load the module. Create a mount point after that: **mkdir /mnt/zip**. Then you need to mount the drive: **mount /dev/sd[x]4 -t vfat /mnt/zip**, [x] being the next available SCSI drive letter i.e., if you have an SCSI hard drive or CD-ROM, it will be /dev/sda, and your ZIP will be /dev/sdb4. The 4 means the partition number, which is always 4 for the preformatted ZIP disk. To make it permanently load at boot, do the following:

```
echo modprobe [module] >> /etc/rc.d/rc.local
```

Also, you should create a mount point for it in /etc/fstab:

```
/dev/sd[x]4 /mnt/zip auto defaults 0 0
```

--Garrett Mickelson, gmickelson@penguincomputing.com

Moving from Microsoft to MySQL on Debian

I'm wondering about the best way to transfer the content (text and graphic files) of an SQL 7.0 database (running on NT Server) into a MySQL database running on Debian GNU/Linux. A related question is how to best migrate ASP pages to a Debian (or other version) Linux platform. Your guidance is greatly appreciated. —Carl Lawson, lawson@oz.net

First off, moving one database to another is not a trivial task. It's like porting a program from one platform to another, there are implementation-specific issues to consider. For example, MySQL does not support some SQL features that Microsoft SQL Server does. The easiest way to port this would be to dump the contents of the SQL Server database to a text file, re-create the tables in MySQL and then use the **mysqlexport** command to read the data into your new database. Another option would be to move the data programmatically, using Perl or Java with ODBC. This would allow you to validate or manipulate the data before inserting it. As for ASP, install Apache with mod_perl support and then install the ASP module. For more information and downloads, go to <http://www.nodeworks.com/asp/>. —Paul Christensen, pchristensen@penguincomputing.com

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

New Products

Heather Mead

Issue #84, April 2001

Aduva Manager, MPEG-2 Encoding Solution ZL-330, Rogue Wave's Large-Scale Objects Solutions and more.

Aduva Manager

Aduva, Inc. introduced Aduva Manager, an internet-based service that automates Linux system management by addressing issues such as upgrades, patches and new hardware and software installations for system reliability and compatibility verification. The downloadable client maintains a local current inventory of all components in the user's system, then updates the user with relevant developments. Features include scheduled actions support, critical issue alerts, encrypted connections and automatic management corrections. Aduva Manager is available 24/7 from the Aduva servers and is offered free of charge to noncommercial users.

Contact: Aduva, Inc., 2595 East Bayshore Road, Palo Alto, California 94303, 650-858-8650, info@aduva.com, <http://www.aduva.com/>.

MPEG-2 Encoding Solution ZL-330

The ZL-330 by Zapex Technologies is an MPEG-2/Dolby encoder for satellite, video server, Digital Video Broadcast (DVB) and cable applications. This encoder simultaneously compresses video and audio into MPEG-2 and Dolby digital streams, then multiplexes them into a single transport stream. The ZL-330 performs all encoding functions and can be integrated with interface cards such as SMPTE-310, TCP/IP or DVB-ASI. General availability of the ZL-330 begins in early May.

Contact: Zapex Technologies, Inc., 2432 Charleston Road, Mountain View, California 94043, 650-930-1300, <http://www.zapex.net/>.



MPEG-2 Board

Rogue Wave's Large-Scale Objects Solutions

To help solve the problem of large distributed environments interacting in real time with comprehensive relational database stores, Rogue Wave Software, Inc. announced a Large-Scale Objects Solution. The Solution is designed to persist objects within relational databases over large, widely dispersed applications. Features include robust object applications (including object persistence mechanisms that respect existing relational models) and transaction journaling to ensure data integrity. Consulting services are available to accompany this Solution.

Contact: Rogue Wave Software, Inc., 5500 Flatiron Parkway, Boulder, Colorado 80301, 800-487-3217 (toll free), <http://www.roguewave.com/>.

briQ from Total Impact

The briQ, a PowerPC-based network appliance computer, has been released by Total Impact. Measuring 5.74 inches wide, 1.625 inches tall and 8.9 inches deep, and weighing 1.85 pounds, the briQ provides developers with a production-ready engine suited for insertion into an industry standard half-height drive bay. The briQ uses either PowerPC G3 or G4 processors and is customizable for applications and products such as firewalls, routers, security devices and web servers. Some of the included specifications include 1MB of L2 cache, 100MHz 64-bit system bus, up to 512MB of SDRAM, dual 10/100 base Ethernet support, up to 40GB hard drive, RS-232 interface, low-power requirements and remote management.

Contact: Total Impact, Inc., 295 Willis Avenue, Suite E, Camarillo, California 93010, 805-987-8704, sales@totalimpact.com, <http://www.totalimpact.com/>.



Front View of briQ

Zend Technologies Launches PHP Product Line

Zend Technologies introduced a PHP technology product line targeting the enterprise PHP market, all available through their on-line subscription service. Products include: Zend Cache, a customizable script-caching module that stores an intermediate code version in the server's memory; Encoder Unlimited, which allows increased web site security and distribution without revealing source code by creating a platform-independent binary file; Zend IDE, a suite of tools for remote debugging, text editing and PHP and HTML highlighting; Zend LaunchPad, which provides quality-assured, updated PHP downloads through a GUI module; and On-line Service Support, a dedicated, web-enabled application available 24/7. Subscriptions are available in commercial and noncommercial versions.

Contact: Zend Technologies Ltd., PO Box 3619, Ramat Gan 52136, Israel, 877-ZEND-USA (toll free), info@zend.com, <http://www.zend.com/>.



ADSL Chip Set

An Asymmetrical Digital Subscriber Line (ADSL) chip set from Agere Systems for residential gateway, home networking and PC equipment is now available. The new offering consists of a peripheral component interconnect (PCI) ADSL network interface card, based on one of Agere's DSP client access chips. The chip set is placed on an interface card, which becomes a network adapter, and PC and residential gateway motherboards communicate with the DSP chip. Specialized communication functions include an adaptation layer for ATMs, network protocols including PPP over Ethernet and ATM, and segmentation and reassembly of ATM packet functions.

Contact: Agere Systems, Inc., 555 Union Boulevard, Room 30L-15P-BA,
Allentown, Pennsylvania 18109-3286, 800-372-2447 (toll free),
docmaster@micro.lucent.com, <http://www.agere.com/>.

Linux Support for HP Printers

At LinuxWorld in January 2001, Hewlett-Packard announced full-feature Linux support for 28 PostScript LaserJet and business inkjet printers. Basic support is also available for 16 non-PostScript LaserJet and inkjet printers. Users can access all device configuration options such as duplexing, tray selection and paper handling options. Installation of a Linux printing system upgrade is necessary for full-feature support, available at <http://www.hp.sourceforge.net/>. All future PostScript HP LaserJet printers will come with full-feature Linux support when introduced to market.

Contact: Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, California 94304-1185, 650-857-1501, <http://www.linux.hp.com/>.

K2 NAS

K2 NAS from Big Storage, Inc. is an enterprise-grade network attached storage device with redundancy features at every level, including integrated network and storage processors. K2 scales from one to forty terabytes and comes with dual redundant server processors, hardware RAID controllers, snapshot software for archiving and journaling filesystems. The system can be plugged directly into the existing network with a quick setup. A web-based graphical interface allows configuration and monitoring from any client, and cross-compatibility supports Windows98/2000, Mac and UNIX.

Contact: Big Storage, Inc., 19 Heron Street, San Francisco, California 94103,
800-864-3789 (toll free), techsales@bigstorage.com, <http://www.bigstorage.com/>.



[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.