

Advanced search

Linux Journal Issue #78/October 2000



Features

Open-Source Intrusion Detection Tools for Linux by Bobby S. Wen

Armed with Linux and Open Source tools, you can even keep an ISP secure.

Securing DNS and BIND by Michael D. Bauer

Decreasing the vulnerability of your DNS server is largely a matter of staying current and private.

Using Postfix for Secure SMTP Gateways by Mick Bauer and Brenno de Winter

Improve your site's e-mail hygiene and make life difficult for spammers and hackers.

Focus: Security by Don Marti

The time for security excuses is over.

Indepth

RTcmix for Linux (Part 1) by Dave Topper

In the first part of this three-part series on real-time audio synthesis we take you through the history and basis of RTcmix.

Apache User Authentication by Ibrahim F. Haddad

A guide to setting up user authentication for the Apache web server running on Linux, using the plaintext file method.

Distance Education Using Linux and the MBone by Kelly Davis, Dr. Tom Miller and Charles Price

There is more to the Internet than sending JPGs. See how Linux and the MBone addresses the needs of distance learning.

Automated Installation of Large-Scale Linux Networks by *Ali Raza Butt and Jahangir Hasan*

Need to load Linux on 100 workstations? Learn some tricks and techniques that could save you days of tedious work.

Graphics: Pick a Card...Any Card by *Matt Matthews*

With graphics capabilities being so important and new cards appearing all the time, you need a scorecard to pick the right one. Here it is ...

BusyBox: A Swiss Army Knife for Linux by *Nicholas Wells*

Learn how to save disk space with this open source tool for embedded systems.

Toolbox

Kernel Korner Contributing to the Linux KernelThe Linux Configuration System by *Joseph Pranevich*

At the Forge Configuring, Tuning, and Debugging Apache by *Reuven M. Lerner*

Cooking with Linux A Few Recipes for Easier Firewalls by *Marcel Gagné*

Columns

Linley on Linux Linux Drives Digital Audio Revolution by *Linley Gwennap*

Focus on Software by *David A. Bandel*

Embedded Systems News by *Rick Lehrbaum*

The Last Word by *Stan Kelly-Bootle*

Reviews

Comparison of Backup Products by *Charles Curley*

Easysoft Data Access Middleware by *Jon Valesh*

Magic Enterprises Edition 8.3 for Linux by *Jon Valesh*

Omnis Studio RAD by *Nick Wells*

Unix Backup and Recovery by *Charles Curley*

LaTeX for Linux by *Ben Crowder*

The XML Handbook 2nd Edition by *Daniel Lazenby*

Securing Linux by *Charles Curley*

Building Linux and OpenBSD Firewalls by *Ralph Krause*

Linux Programming Bible by *Ben Crowder*

Departments

Letters

upFRONT

From the Editor Goodbye Bandits, Hello Security by *Don Marti*

From the Publisher UnixWare and Linux Get Hitched by *Phil Hughes*

Best of Technical Support

New Products

Archive Index

Advanced search

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

Open-Source Intrusion-Detection Tools for Linux

Bobby S. Wen

Issue #78, October 2000

Armed with Linux and open-source tools, you can even keep an ISP secure.

As an ISP, we are the most vulnerable to attack because of the open nature of our networks. Unlike corporate networks, which can limit access, and can backtrack users, we have to continuously monitor for attacks and, more importantly, successful intrusions. We use several open-source security tools to monitor for intrusions and system compromises. These tools have alerted us to attack and allowed us to quickly respond to intrusions from system compromises.

Introduction

System administrators are known to be trained paranoids. They are not only concerned about updating their systems against the latest program vulnerability, hacking tools and kiddie scripts, but they have to be vigilant against the security holes and exploits that are not known or documented. Most good system administrators can protect their systems against low-level crackers by using standard system hardening techniques ("Post-Installation Security Procedures", *Linux Journal*, December 1999). It is the holes that are not known, and the knowledgeable cracker that poses the much greater danger.

Intrusion detection and recovery is a goal of all system security. It usually involves looking for system compromises: checking logs for unusual activity, unusual connections, and alterations in the system files. Of course a system must be secured in order for intrusion detection and recovery to be effective. System administrators would be working double time if they kept finding people breaking into their systems and had to recover from them.

There are many open-source tools available to system administrators to aid in this process.

Intrusion Detection

We use Tiger, Logcheck and Tripwire in cron jobs to continuously monitor the system for incursions and intrusions. They alert us to attacks on our system and allow us to quickly respond to the attack or limit the damage. We also use other security programs, but to allow this article to be manageable, we shall concentrate on programs with stable releases.

tiger-2.2.4

Tiger is a set of scripts that search a system for weaknesses which could allow an unauthorized user to change system configurations, gain root access, or change important system files. System administrators may remember Tiger as the successor to COPS (Computer Oracle and Password System).

Tiger was originally developed at Texas A&M University and can be downloaded from net.tamu.edu/pub/security/TAMU. There are currently several versions of Tiger available: tiger-2.2.3p1, the latest Tiger with updated check_devs and check_rhost scripts; tiger-2.2.3p1-ASC, tiger-2.2.3p1 with contributions from the Arctic Regional Supercomputer Center; tiger-2.2.4p1, tiger-2.2.3 with Linux support; and, TARA, Tiger Analytical Research Assistant, an upgrade to tiger-2.2.3 from Advanced Research Corporation.

Tiger scans the system for cron, inetd, passwd, files permissions, aliases and PATH variables to see if they can be used to gain root access. It checks for system vulnerabilities in inetd, host equivalents and PATH to determine if a user can gain remote access to the system. It also uses digital signatures, using MD5, to determine if key system binaries have been altered.

Tiger is made up of scripts which may be run together or individually. This allows system administrators to determine the best strategy for checking their system depending on system size and configuration. The Tiger script files are available for download at ftp.linuxjournal.com/pub/lj/listings/issue78.

Tiger can be used as an interactive process, or configured to run any or all the scripts using the tigerrc file. In addition, a site file must include the locations of files Tiger may use, such as Crack and Reporter. Sample tigerrc and site files are included in the distribution. We usually start with everything turned on in the tigerrc and the site-sample files, and customize which scripts to run after the initial audit. Tiger can be run by typing `./tiger` in the Tiger directory. Using Tiger as an interactive operation will run the Tiger scripts without regard to the amount of time it takes to complete.

The individual checks can be spread out over time using the tigercron script without cluttering up your crontab file. Tigercron uses the cronrc file to

schedule the tiger scripts in a manner similar to crontab. Tigercron is a little bit different from the conventional crontab process in that tigercron is run in crontab every hour, and then tigercron, using cronrc in the tiger directory, determines which scripts to run. One advantage of tigercron is it is configured to e-mail a security report only if there is a change from the previous scan. This prevents cluttering the system administrator's e-mail with status information and creates a red flag for the sys admin that the system has changed and to take notice of a possible intruder whenever a Tiger e-mail is generated.

Tiger outputs a security report upon completion. The output is detailed and prefixed by cryptic error messages. Fortunately, Tiger comes with a utility to explain error messages, called tigexp. A explanation of each error message can be obtained by running **tigexp -f <report filename>**.

Tiger 2.2.4 has been updated to run with Red Hat Linux using a 2.0.35 kernel. This is great for people who use Red Hat 5.1 and have not upgraded. Systems using other flavors of Linux, or those that have been upgraded will have to update their binary signature file by using the util/mksig script. This allows Tiger binary checks to be updated as new programs are upgraded or installed.

Tiger is an important part of our system integrity monitors. It is proactive in checking for ways root can be compromised and scans for changes in important system files. In the past, we have been alerted to system intrusions because of Tiger.

Logcheck 1.1.1

Logcheck is a script that scans system log files and checks for unusual activity and attacks. This assumes that an intruder has not gained root privileges to a host and cannot alter the log files. One of the big problems with keeping good log files is a lot of information gets collected. Manually scanning log files could take days on large systems. Logcheck streamlines the task of system log monitoring by categorizing the logged information and e-mailing it to the system administrator. Logcheck can be configured to report only the information you want to see and to ignore all information you don't want to see. Logcheck can be downloaded from www.psionic.com/tools/logcheck-1.1.1.tar.gz. It is part of the Abacus Project, an Intrusion Prevention System, however, not all the other components are at stable releases yet. Logcheck is based on a log-checking program called frequentcheck.sh, part of the Gauntlet Firewall Package.

The logcheck.sh script is installed into /usr/local/etc/ along with the keyword files. The script can be downloaded at ftp.linuxjournal.com/pub/lj/listings/issue78.

Logcheck scans the log files for active system attacks first, then security violations, and then unusual system events. Logcheck then e-mails a report to root (see Figure 1).

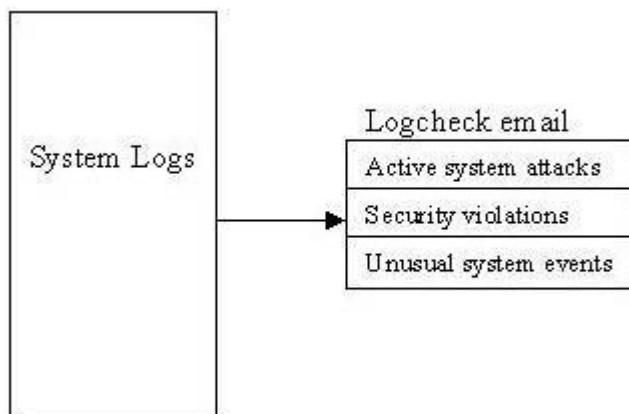


Figure 1. Logcheck Operation

In this sample Logcheck e-mail, I purposely logged on to my system from a host that was not allowed to connect, and then used some common hacking probe commands on my sendmail port. Logcheck detected the attempt to gather userids using sendmail and listed it as an "Active System Attack Alert". It found that a user ftp'ed on to the system in the "Security Violations" section; they may have misspelled their legitimate user ID, or may be a cracker probing the system for user IDs. It also found system trying to connect to us that is not allowed to connect.

Finally, in the "Unusual System Events", anything out of the ordinary that is not specifically ignored is shown. Logcheck creates a .offset file to mark the portions of logs it has scanned to avoid repeating information it has previously processed.

Logcheck is based on the assumption that syslog is configured to log as much information as possible, and, if possible, to log all activity to one file for easier parsing. Parsing only one file ensures that information will not be missed. We logged all our information to /var/log/messages in addition to the standard log files. This requires more disk space, but it makes debugging system problems easier. Logcheck is heavily biased toward Firewall Toolkit and BSDish messages from TCP wrapper. Systems not using Firewall Toolkit or TCP wrappers may need to add keywords for their security/monitoring programs.

Logcheck is useful only to help administrators spot attacks and take precautionary measures, but once an intruder gains access to your system, one of their first steps will be to hide their tracks by disabling logging.

We use Logcheck as a tool to monitor attacks on our system, intrusion activity and system problems. We pay specific attention to individual attacks on our system and cross reference older Logcheck outputs to try to establish patterns of probing and attacks that occur over weeks or months. The distributed or time-spaced probing of our system is considered high-priority because it usually shows a determined or organized attacker with the patience to look for, and take advantage of, new exploits and program holes. We run logcheck in cron daily, and change to an hourly check if we suspect active probing or attacks.

Tripwire

One of the best-known and most useful tools in intrusion detection and recovery is Tripwire. Tripwire creates a signature database of the files on a system, and when run in compare mode, will alert system administrators to changes in the file system. Tripwire is different from Tiger in that it is a dedicated file signature program, which can use multiple hash functions to generate digital signatures.

Tripwire was developed for the Computer Operations Audit and Security Technology (COAST) laboratory at Purdue University. The publicly available version, Tripwire version 1.2, is available from www.cs.purdue.edu/coast/archives Tripwire version 1.2-2 is shipped with Red Hat on the powertools CD as RPMs and is also available from <ftp://ftp.redhat.com/pub/redhat/powertools/>. Tripwire has been available as a commercial product since January 1999 from Tripwire Inc. Tripwire 1.3 is distributed as an Academic Source release. 2.x.x versions are available in binary-only form. Both can be downloaded from www.tripwire.com/downloads/index.cfm for noncommercial use. Tripwire Inc. has announced it will be adopting an open-source model for Tripwire by the third quarter of 2000, but has not yet settled on a licensing scheme.

A great many articles about Tripwire exist, so we will concentrate on using Tripwire as an intrusion detection tool rather than how to use Tripwire.

As I mentioned before, when an intruder penetrates a system, one of the first things they will try to do is hide themselves and their activities from the system. In order to accomplish this, they will commonly substitute system files with Trojan horse binaries—binaries with code added to them—which filter their activities. Common Trojan binaries are TELNET, login, su, FTP, ls, netstat, ifconfig, du, ps, inetd and syslogd. Intruders will also try to create back doors for themselves by adding a user to passwd or a service to inetd. In most cases, they will try to take the /etc/passwd file and run Crack on it hoping to find users with weak passwords.

Intruders will also install tools to allow them to monitor the network or maintain access to a system. They will try to install programs with known vulnerabilities to allow them to gain root access, often in “hidden” files. They may also install network scanners, and probes to collect information and possibly passwords from network traffic sent in clear text.

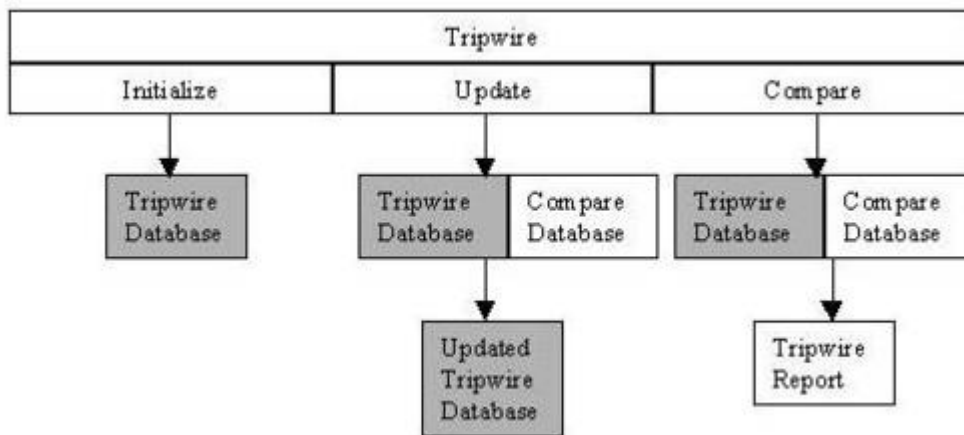


Figure 2. Tripwire Operation

Tripwire as an intrusion-detection tool requires a pristine image database of the system, a snapshot of your system in a known uncompromised state. The best scenario (see Figure 2) is to be able to take a clean snapshot or “fingerprint” when the system is first installed. As programs are installed and upgraded, the system administrator compares the new file signatures to make sure only the programs they worked on are updated to the database and that no other system files were altered.

Capturing the digital signature of the entire file system can create an information overload for the system administrator. Parts of the file system are supposed to be changed with normal operations, so capturing the files in `/var/spool`, `/tmp`, or even `/var/log` can be useless because they are constantly being altered. At the same time, we need to be able to identify files hidden in the system if we detect a compromise. To accomplish the different need of the scans, we create two Tripwire configuration files with two Tripwire databases: one to scan the entire system and one to scan only critical system files.

System administrators have different opinions as to which files are “critical”. We take the view that system binaries and libraries should not change unless we change them. So all `*bin` and `*lib` directories, such as `/bin`, `/sbin`, `/usr/bin`, `/usr/sbin`, `/lib` and `/usr/lib` should be fingerprinted by Tripwire. In addition, all system configuration files in `/etc` should be fingerprinted. The files and directories to be scanned can be configured using `tw.config`. We can redirect the config file and database to be used by using the `-c` and `-d` switches of Tripwire.

Tripwire can be used as an intrusion-detection tool by having it run comparisons as a cron process. Any system files that were altered without the system administrator's knowledge would be a sign that the system has been compromised and that immediate action needs to be taken to limit the damage. Clear signs that a system has been compromised are changes to system files involved in login, access privileges, and process monitoring or accounting.

When an intrusion is detected or suspected, Tripwire can give the system administrator an indication of the extent of the damage by identifying the files that have been changed. If an intrusion has been detected early, the cracker may not have had time to penetrate the system, and may have had time only to start the process by bringing over tools and programs. This is where the fingerprint of the full system is valuable. Using a full scan of the system, hidden files and directories can be identified and investigated. Rather than manually scanning the file system for new directories and files, Tripwire will flag them and output them to the Tripwire report.

A knowledgeable cracker, one who knows about intrusion detection, will commonly try to hide programs in the file structure where they knew the system files would be changed during normal operations, thereby hiding themselves from abridged cron Tripwire comparisons. Places where a system administrator should pay especial attention is /tmp, /dev and /var/spool directories because files are created and deleted during normal operations. Crackers will try to hide their directories by prefixing them with . (dot), so they do not show in a normal ls command.

File integrity checkers should be a part of every system's security; Tripwire is the "best of breed" in this area. It can be used as an alarm to a system penetration and in intrusion recovery.

Intrusion Recovery

When an intrusion has been detected, the system administrator needs to first regain control of the compromised system by disconnecting it from the network. This is to prevent further intrusion and possible Denial of Service attacks on the Internet originating from the compromised host. An image of the system should be backed up to allow for the intrusion to be analyzed and referenced later.

The system must then be analyzed thoroughly by reviewing log files. This is a primary source of information on how, when and where the intrusion occurred. All system binaries and configuration files, including the kernel, need to be verified to make sure they are unaltered. To do this, the system administrator must first insure the system analysis tools themselves are clean and do not

contain Trojans. System data should also be checked to make sure the intruder has not changed them. Intruders may “park” data or programs on the system. This may include programs to be used in other intrusions, and data from other compromised systems. Intruders may also install network sniffers and other monitoring programs in hopes of capturing information which will allow them to access other hosts. Once an intrusion has been detected on one system, all the other systems on the network should also be checked for possible intrusion. The intruder may have used the compromised system to gain access to other hosts on the network, or they may have used other hosts to gain access to the system with the detected intrusion.

System administrators should file an incident report for all hosts compromised with a computer coordination center, such as CERT. Intruders usually use compromised accounts to attack other system. It is difficult or impossible for an individual system to track down the origins of a knowledgeable attacker. However, it is *made* possible through cooperation among system administrators, closing down avenues of attack and access, limiting the attacker to hosts and systems where they can be monitored and identified.

Once an intrusion has been analyzed and reported, then comes the task of recovering from the intrusion. First a clean version of the system should be installed, preferably from the original installation media. If a backup is used, the system binaries should be restored from copies with known clean binaries. The sys admin should take the paranoid stance that the latest backup may contain the altered programs and data and needs to be sure they are not reinstalling bad files.

Once a compromised system has been restored, it must be secured to prevent another intrusion. Steps in hardening a system include disabling all unnecessary services, installing all vendor security patches, consulting CERT and other security advisories, and changing passwords.

Conclusion

Detecting and recovering from an intrusion may actually be the start of a system administrator's security journey. Intrusions only highlight the need for system security. With millions of users on the Internet, one has to assume that, while individually they may pose minimal threat, collectively they are more knowledgeable and have more resources than any system administrator or security program.

Bobby S. Wen (wen@vasia.com) holds two engineering degrees and an MBA. He started playing with Linux in 1994 with a Slackware pre-1.0 distribution and has been addicted ever since. Even though he has a computer for every man, woman, child, and dog at

home, he still has to wait his turn for a computer, because the only computer his children want to play with is the one he is working on. He currently multi-boots Linux, FreeBSD, Solaris, BeOS, Windows 98 and Windows NT.

Listings for [Tiger Script Files](#) and [Active System Attack Alerts](#) are available.

Resources

email: wen@vasia.com

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

[Advanced search](#)

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

Securing DNS and BIND

Michael D. Bauer

Issue #78, October 2000

Decreasing the vulnerability of your DNS server is largely a matter of staying current and private.

In the SANS Institute's recent consensus document "How to Eliminate the Ten Most Critical Internet Security Threats" (www.sans.org/topten.htm), the number-one category of vulnerabilities reported by survey participants was BIND weaknesses. BIND, of course, is the open-source software package that powers the majority of Internet DNS servers. In fact, again according to SANS, over 50% of BIND installations are vulnerable to well-known (and in many cases, old) exploits.

The good news is that armed with the simple concepts and techniques I'm about to describe, you can quickly and easily enhance BIND's security on your Linux (or other UNIX) DNS server. Since our focus here will be security, if you're an absolute BIND beginner you may wish to first start reading the BIND online documentation (see Conclusions, at end) or the first chapter or two of Albitz and Liu's book *DNS and BIND*.

BIND Basics

Having said that, let's begin with a brief look at how the Domain Name Service and BIND work. Suppose someone (myhost.someisp.com in Figure 1) is surfing the Web, and wishes to view the site <http://www.wiremonkeys.org/>. Suppose, also, that person's machine is configured to use the name server "ns.isp.com" for DNS lookups. Since the name www.wiremonkeys.org has no meaning to the routers through which the web query and its responses will pass, the user's web browser needs to learn www.wiremonkeys.org's IP address before attempting the query.

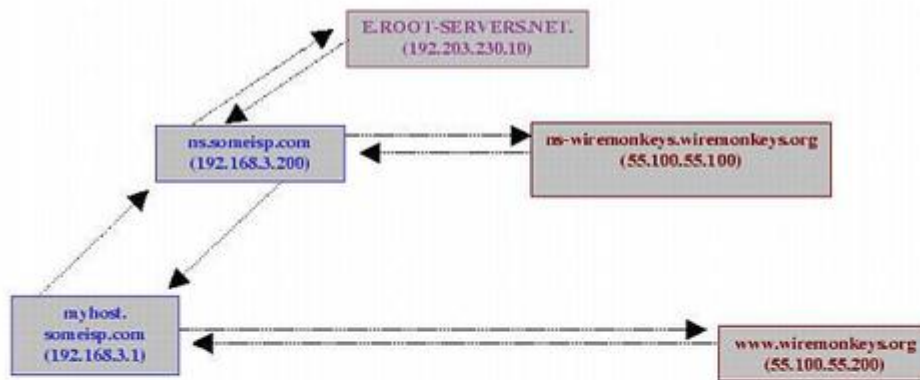


Figure 1: A Simple DNS Query

Figure 1. myhost.someisp.com

First, “myhost” asks “ns” whether it knows the IP address. Since ns.someisp.com isn't authoritative for wiremonkeys.org and hasn't recently communicated for any host that is, it begins a query of its own on the user's behalf. The process of making one or more queries in order to answer other queries is called **recursion**.

ns.someisp.com begins its recursive query by asking a “root name server” for the IP address of some host that's authoritative for the zone wiremonkeys.org. (All Internet DNS servers use a static “hints” file to identify the thirteen or so official root name servers. This list is maintained at <ftp.rs.internic.net/domain> and is called named.root.) In our example, ns asks E.ROOT-SERVERS.NET (an actual root server, with a current IP address of 192.203.230.10), who replies that DNS for wiremonkeys.org is handled by “ns-wiremonkeys.wiremonkeys.org”, with an IP address of 55.100.55.100.

ns then asks ns-wiremonkeys for the IP address of www.wiremonkeys.org. ns-wiremonkeys returns the answer (55.100.55.200), which ns forwards back to myhost.someisp.com. Finally, myhost contacts 55.100.55.200 directly via HTTP and performs the web query.

This is the most common type of name lookup. It and other single-host type lookups are simply called “queries”; DNS queries are handled on UDP port 53.

Not all DNS transactions involve single-host lookups, however. Sometimes it is necessary to transfer entire name-domain (zone) databases: this is called a zone transfer, and it happens when you issue an **ls** command from the nslookup utility, or run **dig**. The main purpose of zone transfers, however, is for name servers that are authoritative for the same domain to stay in sync with each other (e.g., for “master to slave” updates). Zone transfers are handled on TCP port 53.

The last general DNS concept we'll touch on here is caching. Name servers cache all local zone files (i.e., their hints file plus all zone information for which they are authoritative), plus the results of all recursive queries they've performed since their last startup. That is, almost all: each resource record (RRs) has (or inherits their zone-file's default) time-to-live settings. These settings determine how long each RR can be cached before being refreshed.

This, of course, is only a fraction of what one needs to learn in order to fully understand and use BIND. I haven't even mentioned forwarders or reverse lookups. Hopefully, it's enough for the purposes of discussing BIND security.

DNS Security Principles

DNS security can be distilled into two maxims: always run the latest version of your chosen DNS software package, and never provide unnecessary information or services to strangers. Put another way, keep current and be stingy!

This translates into a number of specific techniques. The first is to limit or even disable recursion. Limiting it is easy to do using configuration-file parameters; disabling recursion altogether may or may not be possible, depending on the name server's role.

If, for example, the server is an “external” DNS server with the sole purpose of answering queries regarding its organization's public servers, there is no reason for it to perform lookups of nonlocal host names (which is the very definition of recursion). On the other hand, if a server provides DNS resolution to end users on a local area network (LAN), it definitely needs to recurse queries from local hosts, but can probably be configured to refuse recursion requests, if not *all* requests, from nonlocal addresses.

Another way to limit DNS activity is to use split DNS services (see Figure 2). Split DNS refers to the practice of maintaining both public and private databases of each local name domain (zone). The public zone database contains as little as possible: NS records listing publicly accessible name servers, MX records listing external SMTP (e-mail) gateways, public web servers and other hosts that one wishes the outside world to know about.

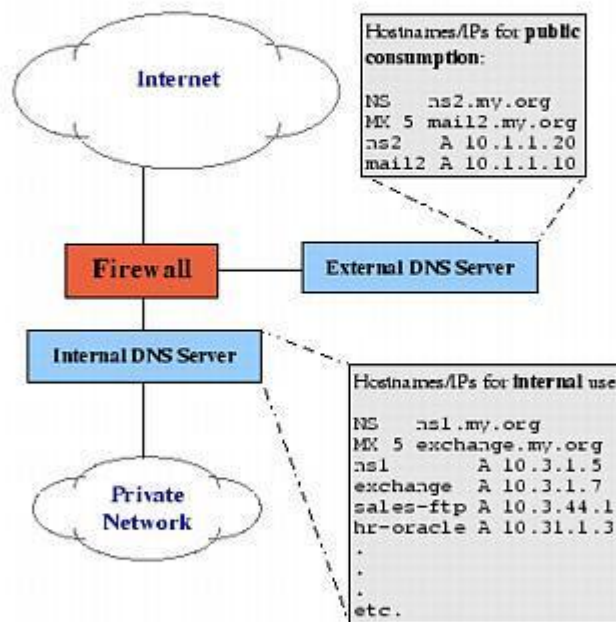


Figure 2: Split DNS

Figure 2. Split DNS

The private zone database may be a superset of the public one, or it may contain entirely different entries for certain categories or hosts. For example, many organizations use a Microsoft Exchange server for internal e-mail, but maintain a completely separate SMTP gateway system to receive mail from the outside world. This is sometimes actually the organization's firewall, or perhaps a dedicated mail server in a DMZ network connected to the firewall but separate from the internal network.

The value of such an architecture should be obvious: compromise of the SMTP gateway does not automatically result in the exposure of internal e-mail to outsiders. Other services commonly split this way are WWW (which separates public web data from intranet data), FTP, and virtually all other TCP/IP services

for which it's desirable to differentiate between public and private data. DNS, however, is arguably the most important service to split, since most other TCP/IP services depend on it.

The other aspect to DNS stinginess is the content of zone files themselves. Even public zone databases may contain more information than they need to. Hosts may have needlessly descriptive names (e.g., you may be telling the wrong people which server does what), or too much or too granular contact information may be given. Some organizations even list individual systems' hardware and software names and versions! Such information is almost invariably more useful to prospective crackers than their intended audience.

Maintaining current software and keeping abreast of known DNS exposures is at least as important as carefully considering actual DNS data. Furthermore, it's easier: the latest version of BIND can always be downloaded for free from <ftp.isc.org>, and information on BIND vulnerabilities is disseminated via not only one, but several mailing lists and newsgroups (some of which are listed at the end of this article).

There's actually a third maxim for DNS security, but it's hardly unique to DNS: take the time to understand and *use* the security features of your software (and of your DNS-registration provider—Network Solutions and other top-level-domain registrars all offer several change request security options, including PGP. Make sure that your provider requires *at least* e-mail verification of all change requests for your zones!).

Selecting and Installing the Appropriate BIND Software

As of this writing, the most current version is 8.2.2, patch level 5. Due to a particularly nasty buffer-overflow problem that can lead to unauthorized root access of vulnerable systems (the "NXT" bug, described in CERT Advisory #CA-99-14) in *all* older 8.2 releases, it is *essential* that anyone using BIND be at least at version 8.2.2P5.

Note that for some time after BIND v.8.1 was initially released in May 1997, many users deliberately continued using BIND v.4 due to its stability (and possibly to postpone having to learn the new configuration-file syntax). In fact, the Internet Software Consortium(ISC) continued to support, and even patch, version 4, releasing BIND v.4.9.7 a full year after 8.1's debut.

However, the ISC no longer recommends that anybody continue using BIND v.4, even v.4.9.7. So it bears repeating: everyone who runs BIND on an Internet server should be running the latest release of version 8.x.

We've established that you need the latest version of BIND. But should you use a pre-compiled binary distribution (such as an RPM), or should you compile it from source? For most users, it's perfectly acceptable to use a binary distribution, provided it comes from a trusted source. Virtually all UNIX variants include BIND with their "stock" installations; just be sure to verify that you indeed have the latest version.

The command to do this with Red Hat Package Manager is **rpm -q -v bind8** if the package has already been installed, or **rpm -q -v -p ./<path & filename of package>** if you have a package file, but it hasn't been installed yet. The rpm package name for BIND is usually "bind8" or "bind".

If you perform this query and learn you have an old (pre-8.2.2p5 version), most package formats support an "upgrade" feature; simply download a more current package version from the Linux distribution web site, and upgrade it using your package manager. To do this with RPM, the command syntax is **rpm -U ./<path & filename of package>**, assuming you don't need special install options. If the above doesn't work, you can try **rpm -U --force ./<path & filename of package>**.

If you can't find a suitable binary distribution, compiling it from source is easy: there is no "configure" script to run, and none of BIND v.8x's Makefiles need be edited. Simply follow the brief instructions in the source's INSTALL file (**make; make install** is all most people need to do).

Starting named: The Padded Cell

It's premature to start **named** (the main process of BIND) just yet. But how you plan to run named will determine how it should be configured. Now is a good point, therefore, to talk about some startup options that can enhance security.

As with all Internet services, it's a good idea to run named in a "padded cell" in which a prospective cracker will be trapped should he/she exploit, for example, some obscure buffer-overflow vulnerability. Three flags make this cell easy to achieve: **-u <username>**, **-g <group name>** and **-t <directory for named to chroot to>**.

The first causes named to run under the specified user name. The second causes named to run under the specified group name, while the third changes ("chroots") the root of all paths referenced by named. Note that when running named chrooted, this new root is applied even *before* named.conf is read. Therefore if, for example, you invoke named with **named -u named -g wheel -t /var/named**, then it will look for named.conf in /var/named/etc rather than in /etc. That is, the default location of named.conf is always /etc, but if named is chrooted to path /other/path, /etc will be translated as /other/path/etc.

The net effect of these three flags (when used properly) is that named's permissions, environment and even file system are severely limited. Should an unauthorized user somehow hijack named, instead of gaining root permissions (prior to BIND v.8, named ran as root) they'll gain the permissions of an unprivileged account. Furthermore, they'll see even *less* of the server's file system than an ordinary user can: directories connected to directory-tree nodes higher than the chroot-point won't, from named's perspective, even exist.

Overview of named.conf

Running **named** in a padded cell is paranoid and elite in itself. But that's just the beginning! BIND 8.x's configuration file, named.conf, with its large number of supported parameters, allows you to control named's behavior with a great deal of granularity.

Consider the example named.conf file as shown in Figure 3.

```
#Figure 3: example named.conf for external-DNS server
acl trustedslaves { 192.168.20.202; 192.168.10.30};

options {
    directory "/";
    listen-on { 168.192.100.254; };
    recursion no; fetch-glue no;
    allow-transfer { trustedslaves; };
};

logging {
    channel seclog {
        file "var/log/sec.log" versions 5 size 1m;
        print-time yes; print-category yes; };
    category xfer-out { seclog; };
    category panic { seclog; };
    category security { seclog; };
    category insist { seclog; };
    category response-checks { seclog; };
};

zone "coolfroods.ORG" {
    type master;
    file "master/coolfroods.hosts";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "master/0.0.27.rev";
};

zone "100.168.192.in-addr.arpa" {
    type master;
    file "master/48.42.208.rev";
};
```

Figure 3. Example named.conf File

The hypothetical server whose configuration file is represented here is an external DNS server. Since its role is to provide information to the outside world about coolfroods.org's publicly accessible services, it has been configured without recursion. In fact, it has no "." zone entry (i.e., no pointer to a hints file), so it knows nothing about and cannot even learn about hosts not described in its local zone files. Transfers of its local zone databases are restricted by IP address to a group of trusted slave servers, and logging has been enabled for a variety of event types.

So how do we do these and even more nifty and paranoid things with **named.conf**?

Useful named.conf Parameters: acl{} Sections

Although strictly optional, Access Control Lists (acls) provide a handy means of labeling groups of IP addresses and networks. And since we're paranoid, we definitely want to restrict certain actions and data by IP address.

An acl may be declared anywhere within named.conf, but since this file is parsed from top to bottom, each acl must be declared before its first instance in a parameter. Thus, it makes sense to put acl definitions at the top of named.conf.

The format for these is simple:

```
acl acl_name { IPAddr1;  
IPAddr2; ...etc. };
```

Note that the IP address list can contain either complete IP addresses in the form x.x.x.x or network addresses in the form x.x.x/24, x.x/16, etc. Now, each time named.conf is read, the parser will substitute all instances of the acl's name (that occur after its definition) with its corresponding IP address list.

Global Options: The options{} Section

The next thing to add is a list of global options. Some of the parameters that are valid for this section can also be used in zone sections; be aware that if a given parameter appears both in options{} and in a zone section, the zone version will supercede the global setting as it applies to that zone. In other words, the zone-section values of such parameters are treated as exceptions to the corresponding global values.

Listing 2 shows some useful parameters that can be used in options{}.

Listing 2. options {} Listing

Logging

In addition to global options, we definitely want to set some logging rules. By default, named doesn't log much more than a few startup messages (such as errors and zones loaded), which are sent to the syslog daemon (which in turn writes them to `/var/log/messages` or some other file). To log security events, zone transfers, etc., you need to add a **logging** section to `named.conf`.

The **logging** section consists of two parts: one or more **channel** definitions (each of which defines a place to send log information) followed by one or more **category** sections (wherein each event type you wish to track is assigned one or more channels). Channels usually point to either files or the local syslog daemon, and categories are actually pre-defined; that is, you choose from a set of pre-defined categories, specifying in each case what is to be done with event messages from that category.

Channel definitions take the following format: **channel channel-name** **{filename syslog syslog-type | null}; print-time [yes | no]; print-category[yes | no];};**

Note that *filename* by default is put in named's working directory, but a full path may be given (that is assumed to be relative to the chrooted directory, if applicable).

Category specifications are much simpler:

category category-name {channel-list ;};

Note that as with IP address lists, the channel list is semicolon-delimited and must contain channels defined in a prior **channel** statement. See the BIND Operator's Guide (BOG) for the full list of supported categories; suffice it to say that **xfer-out**, **security**, **load**, **os**, **insist**, **panic** and **maintenance** are usually of interest to the security-conscious DNS administrator.

A Note on Caching-Only Name Servers

"Caching-only" name servers, which are not authoritative for any zones (i.e., are neither master, slave, nor even stub for anything), are inherently simpler and therefore easier to secure than other kinds of DNS servers. Little of what follows will apply when setting up a caching-only server.

zone{} Sections

The last type of named.conf section we'll examine here is the **zone{} section**. Like **options{}** , there are many additional parameters besides those described below; see the BOG for more information.

The three parameters most useful in improving zone-by-zone security are:

```
allow-update { IP/acl-list ; };
allow-query IP/acl-ist ; };
allow-transfer IP/acl-list ; };
```

allow-update lists hosts that may submit Dynamic DNS updates for the zone; **allow-query** specifies which hosts may even submit simple DNS queries; and **allow-transfer** restricts who may download entire zone files. Note that all three of these parameters may be used in either/both **zone{} sections** and/or the **options{} section**, with zone-specific settings overriding global settings.

Zone File Security

Our secure DNS service, trapped in its padded cell and very particular about what it says to whom, is shaping up nicely. But what about the actual zone databases?

The good news here is that since our options are considerably more limited than with named.conf, there's less to do. The bad news is that there's at least one type of Resource Record that's both obsolete and even dangerous, and must be avoided by the security-conscious.

Here's a sample zone file for the hypothetical domain "boneheads.com" (see Figure 4.)


```

@ IN SOA  cootie.boneheads.com. hostmaster.boneheads.com.
(
    2000060215          ; serial
    10800              ; refresh (3H)
    1800               ; retry (30m)
    1209600           ; expiry (2w)
    432000 )          ; RR TTL (12H)
IN      NS      ns.otherdomain.com.
IN      NS      cootie.boneheads.com.
IN      MX 5    cootie.boneheads.com.
blorp   IN      A      10.13.13.4
cootie  IN      A      10.13.13.252
cootie  IN      HINFO   MS Windows NT 3.51, SP1
@       IN      RP      john.smith.boneheads.com. dumb.boneheads.com.
dumb    IN      TXT     "John Smith, 612/231-0000"

```

Figure 4: Sample Zone File

Figure 4. Sample Zone File

The first thing to consider is the Start-of-Authority (SOA) record. In the above example, the serial number follows the convention `yyyymmdd###`, which is both convenient and helps security, as it reduces the chances of accidentally loading an old (obsolete) zone file—the serial number serves as both an index and a time stamp.

The refresh interval is set to three hours, a reasonable compromise between bandwidth conservation and paranoia. That is, the shorter the refresh interval, the less damage a DNS-spoofing (cache-poisoning) attack can do, since any “bad records” propagated by such an attack will be corrected each time the zone is refreshed.

The expiry interval is set to two weeks. This is the length of time the zone file will still be considered valid, should the zone's master stop responding to refresh queries. There are two ways a paranoiac might view this parameter. On one hand, a long value ensures that should the master server be bombarded with denial-of-service attacks over an extended period of time, its slaves will continue using cached zone data and the domain will continue to be reachable (except, presumably, for its main DNS server!). But on the other hand, even in the case of such an attack, zone data may change, and sometimes old data causes more mischief than no data at all.

Similarly, the Time to Live interval should be short enough to facilitate reasonably speedy recovery from an attack or corruption, but long enough to prevent bandwidth cluttering. (The TTL determines how long the individual

zone's Resource Records may remain in the caches of other name servers retrieving them via queries.)

Our other concerns in this zone file have to do with minimizing the unnecessary disclosure of information. First, we want to minimize aliases ("A records") and canonical names ("CNAMEs") in general, so that only those hosts who need to be are present. (Actually, we want split DNS, but when that isn't feasible or applicable, we should still try to keep the zone file sparse.)

Second, we want to minimize the amount of (recursive) glue-fetching that goes on. This occurs when a requested name-server (NS) record contains a name whose IP address (via an A record) is not present on the server answering the NS query. In other words, if server X knows that Y is authoritative for domain WUZZA.com but X doesn't actually know Y's IP address, life can get weird: this scenario paves the way for DNS-spoofing attacks. Therefore, if you really want to eliminate all recursion (and I hope you do by now), make sure none of your Resource Records require recursive glue-fetching, and then set the "fetch-glue" option to "no".

Finally, we need to use RP and TXT records judiciously if at all, but must never, ever put any meaningful data into an HINFO record. RP, or Responsible Person, is used to provide the e-mail address of someone who administers the domain. This is best set to as uninteresting an address as possible, e.g., "information@wuzza.com" or "hostmaster@wuzza.com". Similarly, TXT records contain text messages that have traditionally provided additional contact information (phone numbers, etc.) but should be kept only specific enough to be useful, or better still, omitted altogether.

HINFO is a souvenir of simpler times: HINFO records are used to state the operating system, its version, and even hardware configuration of the hosts to which they refer! Back in the days when a large percentage of Internet nodes were in academic institutions and other open environments (and when computers were exotic and new), it seemed reasonable to advertise this information to one's users. Nowadays, HINFO has no valid use on public servers, other than obfuscation (i.e., intentionally providing false information to would-be attackers). In short, don't use HINFO records!

Returning to Figure 3, then, we see that the last few records are unnecessary at best and a cracker's gold mine at worst. And although we decided the SOA record looks good, the NS record immediately following points to a host on another domain altogether—remember, we don't like glue-fetching, and if that's the case here, we may want to add an A record for ns.otherdomain.com.

Advanced BIND Security: TSIGs

Whew! We've looked at BIND security from several important angles, but we haven't even mentioned cryptographic controls. Actually, the Secure DNS protocol ("DNSSEC", described in RFC 2535) merits an article of its own. This set of extensions to DNS provides a means of cryptographically signing zone-transfer and query transactions, including the secure exchange of all necessary key data. Since DNSSEC hasn't yet been widely implemented (and in fact isn't even fully supported in BIND v.8x), we'll limit our discussion of it to the use of Transaction Signatures (TSIGs).

Suppose you wish to sign all zone transfers between the zone's master and a slave. You would do the following:

1. create a key for the zone
2. on each server, create a **key{}** entry in named.conf containing the key
3. on each server, create a **server{}** entry in named.conf for the other server which references the key declared in step (2).

Step one is most easily done with BIND's **dnskeygen** command. To create a 512-bit signing key that can be used by both master and slave, type **dnskeygen -H 512 -h -n <desired_keyname>**. The output will be saved in two files named something like **Kdesired_keyname.+157+00000.key** and **Kdesired_keyname.+157+00000.private**. In this case, the key-string in both files should be identical; it will look something like
"ff2342AGFASsdfsfa55BSopiue/-2342LKJdJlkjVVVvfjweovzp2OIPOTXUEdss2jsdfAAalskj==".

Steps two and three consist of creating entries similar to the following in named.conf on each server (substituting "desired_keyname" below with whatever you wish to name the key—this string must be the same on both servers!):

```
key desired_keyname {
    algorithm hmac-md5;
    secret "<insert key-string from either keyfile here>";
}
server <IP address of other server> {
    transfer-format many-answers;
    # (send responses in batches rather than singly)
    keys { desired_keyname; };
};
```

Note that **key{}** statements must always precede any other statements (e.g., **server{}** statements) that refer to them. A logical place to put key-server statements is between **options{}** and zone statements.

Now all you need to do is restart named (via a **kill -HUP** or **ndc restart**) on both servers. *Voilà!* You are now on the cutting edge of DNS security!

Conclusion

The guidelines and techniques we've covered here should give you a good start on securing your DNS server(s). For a more in-depth understanding of these techniques, I strongly recommend BIND's on-line version of its Operators' Guide (included in most binary distributions, or available separately from <http://www.isc.org/>). This is among the most useful documents provided in any OSS package. Another excellent source of BIND security information is Liu's "DNS Security" slide show (available from him in PDF format—see below).

Equally important, every BIND user should subscribe to at least one security-advisory e-mail list. CERT is my personal favorite, since it's timely enough to be useful but low-volume enough to not be a nuisance. And at your earliest convenience, you should look up and read the CERT advisories listed below—understanding the threats is an essential part of good security.

Resources



email: mick@visi.com

Mick Bauer (mick@visi.com) is security practice lead at the Minneapolis bureau of ENRGI, a network engineering and consulting firm. 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.

[Advanced search](#)

Using Postfix for Secure SMTP Gateways

Mick Bauer

Brenno de Winter

Issue #78, October 2000

Improve your site's e-mail hygiene and make life difficult for spammers and hackers.

E-mail is easily the most popular and important Internet service today, which has made it a popular target of cyber-criminals and spam-happy miscreants. Adding to the problem is the inescapable reality that configuring sendmail, the most commonly used Mail Transfer Agent (MTA), is complicated, nonintuitive and easy to get wrong.

Wietse Venema, intrepid developer of TCP wrappers and co-creator of SATAN, has come through for us again: his program, **postfix**, provides an alternative to sendmail that is simpler in design, more modular, easier to configure and less work to administer. Equally important, it's been designed with scalability, reliability and sound security as fundamental requirements.

This article is intended to bring you up to speed quickly on how to use postfix on your network as a secure means of receiving e-mail from and delivering it to Internet hosts. In particular we'll focus on deploying postfix on firewalls, in DMZs and in other settings in which it will be exposed to contact with untrusted systems.

Is sendmail really that bad? That depends on what you need it to do—the learning curve may not be justified if your e-mail architecture is simple. But sendmail is unquestionably an extremely powerful, stable and widely deployed application that isn't going away anytime soon, nor should it. In fact, The Paranoid Penguin will probably feature a sendmail article some time in the next few months.

Background: Mail Transfer Agents

Both sendmail and postfix are Mail Transfer Agents. MTAs move e-mail from one host or network to another. These are in contrast to Mail Delivery Agents, which move mail within a system (i.e., from an MTA to a local user's mailbox, or from a mailbox to a file or directory). In other words, MTAs are like the mail trucks (and airplanes, trains, etc.) that move mail between post offices; Mail Delivery Agents are like the letter-carriers who distribute the mail to their destination mail boxes.

In addition to MTAs and MDAs, there are also various kinds of e-mail readers, including POP, POP3, and IMAP clients for retrieving e-mail from remote systems. These are also known as Mail User Agents, or MUAs. (There is no real-life simile for these, unless your mail is handed to you each day by a minion whose sole duty is to check your mail box now and then!) But we're not concerned with these or with MDAs, except to mention how they relate to MTAs.

By the way, if you still use UUCP, it's supported in postfix (and continues to be in sendmail, too); most MTAs support a variety of delivery "agents", almost always UUCP and SMTP at the very least. Still, for the remainder of this article we'll assume you're interested in using postfix for SMTP (Simple Mail Transfer Protocol) transfers.

SMTP Gateways and DMZ Networks

One very common use of SMTP, especially in organizations which use *other* e-mail protocols internally, is on an Internet e-mail gateway. Since SMTP is the *lingua franca* for Internet e-mail, there must be at least one SMTP host on any network that needs to exchange e-mail over the Internet. In such a network, the SMTP gateway acts as a liason between non-SMTP mail servers on the inside and SMTP hosts on the outside.

This "liason" functionality in and of itself isn't as important as it once was; the current versions of Microsoft Exchange, Lotus Notes, and many other non-SMTP-based e-mail server products have no problem communicating with SMTP servers directly. But there are still reasons to have all inbound (and even outbound) e-mail arrive at a single point, the chief reason being security.

There are two main security benefits to using an SMTP gateway. First, it's much easier to secure a single SMTP gateway from external threats than it is to secure multiple internal e-mail servers. Second, separating Internet mail from internal mail allows one to move Internet mail transactions off the internal network entirely. The logical place for an SMTP gateway is in a DMZ

("Demilitarized Zone") network, separated from both the Internet and the internal network by a firewall.

As with DNS, FTP, WWW and any other publicly accessible service, the more protection you can place between potential hackertargets and your internal network, the better. Adding an extra NIC to your firewall, keeping public services in a separate network, is one of the cheapest and most effective ways of doing this—as long as you configure the firewall to carefully restrict traffic to/from the DMZ). It's also good risk management; in the (hopefully) unlikely event that your web server, for example, is compromised, it won't become nearly as convenient a launch pad for attacks on the rest of your network.

(For additional information on the DMZ technique of firewalling, see the article [Securing DNS and BIND](#), page 92 of this issue.)

Thus, even organizations with only one e-mail server should still consider adding an SMTP gateway, even if that e-mail server already has SMTP functionality.

But what if your firewall *is* your FTP server, e-mail server, etc.? Although the use of firewalls for any service hosting is scowled upon by the truly paranoid, this is common practice for very small networks (e.g., home users with broadband Internet connections). And, in this foul-weather paranoid's opinion, BIND and postfix pose much less of an exposure for a firewall than other service applications.

For starters, DNS and SMTP potentially involve less direct contact between untrusted users and the server's file system. (I say "potentially" because it's certainly possible, with badly written or sloppily configured software, to create extremely insecure DNS and SMTP services.) In addition, both BIND and postfix have "chroot" options and run as unprivileged users, two features that help reduce the danger of either service being used to somehow gain root access (we'll discuss both of these options in depth shortly.)

Postfix Architecture: How Does Postfix Work?

To understand how postfix works, it's useful to consider its background. The main purpose for postfix's existence is sendmail's complexity. Postfix is a full-featured MTA, and therefore its core functions are the same as any other's. But postfix was written with unusual attention to:

- **Security.** Postfix was designed with security as a fundamental requirement rather than as an afterthought. It's obvious that Mr. Venema has taken the lessons of history (as chronicled by CERT, bugtraq, et al.) very much to heart. For example, the system doesn't trust any data,

regardless of its source. And with least privilege in a chrooted jail (see below), risks are reduced. Furthermore, protective measures against buffer overflows and other user-input attacks have been implemented. If something still fails, postfix's protection mechanism tries to prevent any of the processes under its control from gaining rights they shouldn't have. Since postfix is comprised of many different programs that function without a direct relationship to each other, if something goes wrong, the chance that such a problem can be exploited by an attacker is minimized. Of course, we all know that no system is 100% secure; the goal must be to minimize and manage risks. Postfix is definitely engineered to minimize security risks.

- **Simplicity and compatibility.** Postfix has been written in such a way that setting it up "from scratch" can take as little as five minutes. When you want to replace sendmail or other MTAs, it's even better: postfix by default can use the old configuration files!
- **Robustness and stability.** Postfix was written with the expectation that certain components of the mail network (the Local Area Network, the Internet uplink, the local interfaces, etc.) will occasionally fail. By anticipating things that can go wrong at either end of any given transaction, postfix is capable of keeping the server up and running in many (if not most) circumstances. If, for instance, a message cannot be delivered, it is scheduled to be delivered later, without immediately initiating a continuous retry.

A key contributor to the stability and the speed of postfix is the intelligent way in which it queues mail. Postfix uses four different queues, each one of which is handled differently (see Figure 1):

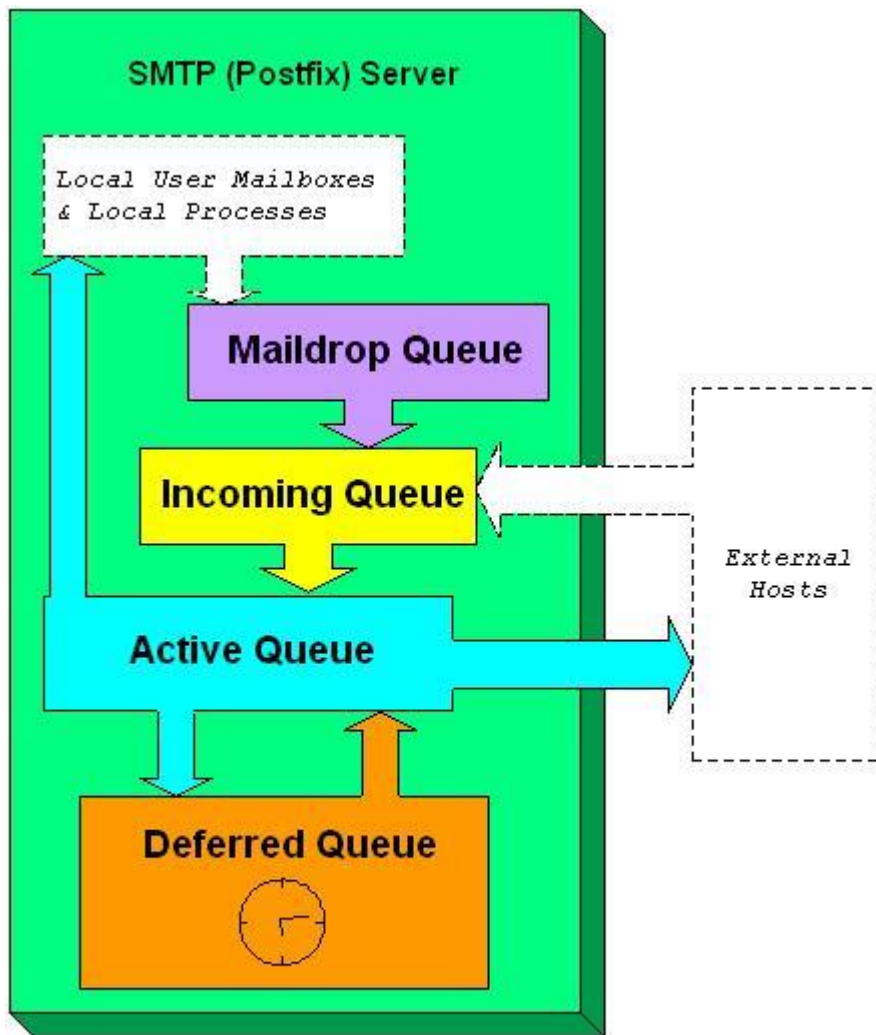


Figure 1: Postfix Queues

Figure 1. Postfix Queues

- **Maildrop queue.** Mail that is delivered locally on the system is accepted in the Maildrop queue. Here, the mail is checked for proper formatting (and fixed if necessary) before being handed to the Incoming queue.
- **Incoming queue.** The Incoming queue receives mail from other hosts, clients or the Maildrop queue. As long as e-mail is still arriving and as long as postfix hasn't really handled the e-mail, this queue is the place where the e-mails are kept.
- **Active queue.** The Active queue is the queue that is used to actually deliver messages and therefore has the greatest potential risk of something going wrong. This queue has a limited size, and messages will be accepted only if there is space for them. That means e-mail in the Incoming and Deferred queues have to wait until the Active queue can accept them.
- **Deferred queue.** E-mail that cannot be delivered is placed in the Deferred queue. This prevents the system from continuously trying to deliver e-mail and keeps the Active queue as short as possible in order to give newer

messages priority. This also enhances stability. If the MTA cannot reach a domain, **all** the e-mail for that domain is placed in the Deferred queue, so that those messages will not needlessly monopolize system resources. Retry is scheduled with an increasing waiting time. When the waiting time expires, the e-mail is again placed in the Active queue for delivery; the system keeps track of retry history.

Postfix for the Lazy: A Quick and Dirty Startup Procedure

And now the part you've been waiting for (or have skipped directly to): postfix setup. Like sendmail, postfix uses a ".cf" text file as its primary configuration file called **main.cf**. However, ".cf" files in postfix use a simple "parameter=\$value" syntax. What's more, these files are extremely well commented and use highly descriptive variable names.

In fact, if your e-mail needs are simple enough, it's probably possible for you to figure out much of what you need to know by editing **main.cf** and reading its comments as you go.

For many users, this is all one needs to do to configure postfix on an SMTP gateway:

1. Install postfix from a binary package via your local package tool (rpm, etc.) or by compiling from source and running postfix's **INSTALL.sh** script.
2. Open /etc/postfix/main.cf with the text editor of your choice.
3. Uncomment and set the parameter **myhostname** to equal your server's fully qualified domain name (FQDN), e.g., "**myhostname = buford.dogpeople.org**".
4. Uncomment and set the parameter **mydestination** as follows, assuming this is the e-mail gateway for one's entire domain:

```
mydestination = $myhostname, localhost.$mydomain, $mydomain
```

NOTE: Enter the above line verbatim.

1. Save and close main.cf.
2. If desired, add a line to /etc/aliases diverting root's mail to a less-privileged account, e.g., **root: mick**. This is also the place to map aliases for users who are served by internal mail servers (for example, **mick.bauer: mbauer@secretserver.dogpeople.org**). When you are done editing and/or adding aliases, save the file and enter the command **newaliases** to convert it into a hash database.
3. Execute the command **postfix start**.

What did we just achieve? In only four steps, we installed, configured and started SMTP services for our machine and its local name-domain. If this machine is a firewall or an SMTP gateway on a firewall's DMZ network, it can now be used by local users to route outbound e-mail, and can be pointed to by our domain's "MX" DNS record (i.e., it can be advertised to the outside world as a mail server for e-mail addressed to our domain). We've also told it to directly process (rather than forward) mail addressed to local hosts. Pretty good return on the investment of about five minutes' worth of typing, no?

(NOTE: While this may be enough to get postfix working, it is **not** enough to secure it. Don't stop reading yet!)

The Quickness and Dirtiness Explained

As cool as that was, it may not have been enough to get postfix to do what needs to be done for your network. And even if it was, it behooves you to dig a little deeper: ignorance nearly always leads to bad security. Let's take a closer look at what we just did, and then move on to even niftier postfix tricks.

First, why did so little information need to be entered in main.cf? The only thing we added to it was our fully qualified domain name. In fact, depending on how your machine is configured, it may not have even been necessary to supply *that!*

This is because postfix uses system calls such as **gethostname** to glean as much information as possible directly from your kernel. If given the fully qualified domain name of your host, it's smart enough to know that everything past the first "." is your name-domain, and it sets the variable **mydomain** accordingly.

You may need to add additional names to **mydestination** if your server has more than one FQDN (that is, multiple "A" records in your domain's DNS). For example, if your SMTP gateway doubles as your public FTP server, and thus has the name "ftp" associated with it in addition to its normal host name, your **mydestination** declaration might look something like this:

```
mydestination = $myhostname, localhost.$mydomain, ftp://www.$mydomain, $mydomain
```

It's important that **any name** by which your server can be legitimately referred to is contained in this line.

There were two other interesting things we did in the "quick and dirty" procedure. One was to start postfix with the command **postfix start**. Just as BIND uses **ndc** to control the various processes that comprise BIND, the **postfix** command can be used to manage postfix. Like BIND, postfix is actually a suite of commands, daemons and scripts rather than a single monolithic program.

The most common invocations of the **postfix** command are **postfix start**, **postfix stop** and **postfix reload**. Start and stop are obvious; reload causes postfix to reload its configuration files without stopping and restarting. Another handy one is **postfix flush**, which forces postfix to immediately attempt to send all queued messages. This is particularly useful after changing a setting that you think may have been causing problems—in the event that your change worked, all messages delayed by the problem go out immediately. They'd go out regardless, but not as quickly.

The other thing we did was to add a line to `/etc/aliases` to divert root's e-mail to an unprivileged account. This is good healthy paranoia: we don't want to have to log in as the superuser for mundane activities such as viewing system reports, which are sometimes e-mailed to root. Be careful, however: if your unprivileged account uses a `“.forward”` file to forward your mail to some other system, you may wind up sending administrative messages over public bandwidth in clear text!

Aliases Revealed

As alluded to in the quick and dirty procedure, aliases are also useful for mapping e-mail addresses for users who don't actually have accounts on the SMTP gateway. This practice has two main benefits. First, most users prefer meaningful e-mail names and short host /domain names, e.g., `“john.smith@acme.com”` rather than `“jsmith023@mail77.midwest.acme.com”`. Second, you probably don't want your users connecting to and storing mail on a publicly accessible server. Again, common sense tells us that any server the unwashed masses can commune with must be kept at arm's length. The greater the separation between public servers and private servers, the better. (And don't forget, POPmail passwords are transmitted in clear text!)

Still another use of **aliases** is the maintenance of mailing lists. An alias can point to not only an address or comma-separated list of addresses, but also to a mailing list. This is achieved with the **:include:tag**—without this, postfix will **append** mail to the file specified rather than using the file to obtain recipients. (This is a feature, not a bug; it's useful sometimes to write certain types of messages to a text file rather than to a mailbox.)

Here's part of an example alias file that contains all of these types of mappings:

```
postmaster:    root
mailer-daemon: root
hostmaster:   root
root:         bdewinter
mailguys:     bdewinter,mick.bauer
mick.bauer:   mbauer@biscuit.stpaul.dogpeople.org
clients:     :include:/etc/postfix/clientlist.txt
spam-reports: /home/bdewinter/spambucket.txt
```

One warning: if an alias points to a different mail server, that server must belong to a domain for which the SMTP gateway is configured to relay mail (i.e., either that server's FQDN or its domain must be listed in the **mydestination** declaration in main.cf).

Don't forget to run either **newaliases** or, hipper still, **postalias /etc/aliases** anytime you edit aliases. The **postalias** command is hipper because it can accept **any** correctly formatted alias file as its input. Both commands compress the alias file into a database file that can be searched repeatedly and rapidly each time a destination address is parsed; neither postfix nor sendmail directly use the text version of **aliases**.

If you have a large number of users and/or internal mail servers, alias-file updates lend themselves to automation, especially via Secure Shell (ssh) and Secure Copy (scp). Using scp with null-passphrase RSA (or DSS/El Gamal) keys, your internal mail servers can periodically copy their local alias files to the SMTP gateway, which can then merge them into a new /etc/aliases followed by **postalias /etc/aliases**. (Unfortunately, telling you exactly **how** to use scp/ssh is beyond the scope of this article.) This practice is especially useful in large organizations where different people control different mail servers: day-to-day e-mail account administration can be kept decentralized.

Keeping out Unsolicited Commercial E-mail

Junk mail is one of the most common and annoying types of e-mail abuse. Postfix offers protection against UCE (Unsolicited Commercial E-mail) via a couple of settings in main.cf. Some caution is in order, however: there's a fine line between spam and legitimate dissemination, and it's entirely possible that even modest UCE controls will cause some legitimate (i.e., desired) mail to be dropped.

Having said that, for most sites this is an acceptable risk (avoidable, too, through end-user education), and we recommend that at a *minimum*, you set the following in main.cf:

- **smtpd_recipient_limit**. This setting indicates how many recipients may be addressed in the header of a single message. Normally, such a number should not exceed something like 500. It would be extreme to receive an e-mail that has 500 recipients and was not being sent to a mailing list.
- **smtpd_recipient_restrictions**. Not every e-mail that arrives at your server should be accepted. This parameter instructs postfix to check each message's recipient-address base on one or more criteria. One of the easiest to maintain is the **access** database. This file lists domains, hosts, networks and users who are allowed to receive mail from your server. To

enable it: (1) set **check_recipient_access = hash:access**; (2) create `/etc/postfix/access` (do a **man 5 access** for format/syntax); and (3) run **postmap hash:/etc/postfix/access** to convert the file into a database. Repeat step (3) each time you edit `/etc/postfix/access`.

- **smtpd_sender_restrictions**. By default postfix will accept SMTP connections from everybody, potentially exposing your server to SMTP relaying, a method often used for UCE perpetrators who wish to hide their identities by “bouncing” their messages off unsuspecting SMTP relayers. If this occurs, it's possible and even likely that other servers will reject e-mail from your domain(s). Other protection mechanisms lie in the fact that it is always wise to check the sender against DNS. Although this costs some performance, it makes it harder to send the information from a faulty sender e-mail address. See the file `/etc/postfix/sample-smtpd.cf` for a list of possible list options for this parameter. Note that **hash:access** is one of them; the **access** database can be used not only to allow/disallow particular recipients, but senders as well. For a complete list of anti-UCE parameters and their exact syntax see `/etc/postfix/sample-smtpd.cf`.

Hiding Internal E-mail Addresses by Masquerading

In order to prevent giving out information that serves no purpose to legitimate external parties, it is wise to set in the `main.cf` file the parameter **masquerade_domains = \$mydomain** (remember, “**\$mydomain**” refers to a variable). If you wish to make an exception for mail sent by “root” (probably a good idea), you can set the parameter **masquerade_exceptions = root**. This will cause internal host names to be stripped from FQDSes in “From” addresses of outbound messages.

Running Postfix in a chroot Jail

Now we come to one of the groovier things we can do to secure postfix: running it in a “chroot jail”. **chroot** is a UNIX command that confines the “chrooted” process to a specified directory; that directory becomes “/” for that process. This usually requires you to first create copies of things needed by the process but normally kept elsewhere. For example, if the process looks for “`/etc/mydaemon.conf`” upon startup but is being chrooted to “`/var/mydaemon`”, the process will actually look for “`/var/mydaemon/etc/mydaemon.conf`”.

The advantage to chrooting should be obvious: should a chrooted-postfix process become hijacked somehow, the attacker will find himself in a “padded cell” from which (hopefully) no sensitive or important system files or data can be accessed. This isn't a panacea, but it significantly increases the difficulty of exploiting postfix.

Happily, the preparations required to chroot postfix are provided in a subdirectory of the postfix documentation called “examples”. These files aren't really shell scripts: they're suggested sequences of commands.

Better still, some binary distributions of postfix have installation scripts that automatically make these preparations for you after installing postfix. In SuSE, for example, the postfix RPM package runs a script that creates a complete directory tree for postfix to use when chrooted (etc, usr, lib, and so forth) in /var/spool/postfix, with the appropriate ownerships and permissions.

In addition to “provisioning” postfix's chroot jail, you need to edit /etc/postfix/master.cf to toggle the postfix dæmons you wish to run chrooted (i.e., put a “y” in the “chroot” column of each dæmon to be chrooted). Do **not**, however, do this for dæmons whose “command” column indicates that they are of type “pipe” or “local”. Some binary-package distributions toggle the appropriate dæmons to chroot automatically during postfix installation (again, SuSE does).

After configuring the chroot jail and editing master.cf, all you need to do is start postfix the way you normally would: **postfix start**. Postfix's master process handles the actual chroot-ing.

Conclusion

That's more than enough information to get you started. May your mail arrive promptly and the spamming filth stay out!

Resources



Mick Bauer is security practice lead at the Minneapolis bureau of ENRGI, a network engineering and consulting firm. 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 sent to mick@visi.com.



Brenno de Winter, 28, is the Linux-focused president of De Winter Information Solutions. He started programming at the age of nine. In his daily routine he is involved in UNIX/Linux, databases, security, telephony-over-IP presentations, consulting and training. He's active in the Polder Linux User Group, has contributed to several GPL projects, including GnuPG, MySQL and TWIG, and is in the process of creating a brand-new project himself as well.

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

[Advanced search](#)

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

[Advanced search](#)

Focus: Security

Don Marti

Issue #78, October 2000

If you read nothing else this month, read Mick Bauer's article about securing your name server.

If you read nothing else this month, read Mick Bauer's article about securing your name server (see page 92). Vulnerabilities in old versions of the name server software BIND are the number-one security problem on Linux systems. If you don't want to wake up to a mailbox full of complaints about script kiddies conducting denial-of-service attacks from your system, do what Mick says. Now. You can read the rest of this any time.

You're back? Good. Now that your name server is secure, we have some other security HOWTO articles for you to read in this issue, too. Your mail server might be another target for an attack. Mick has another helpful security measure: put a SMTP gateway between the outside world and the feature-rich server where the user's mail lives. Especially if you have to support a proprietary mail server and can't count on security fixes, this kind of "bastion host" for mail is better than relying on a firewall that has only a minimal understanding of what mail is.

Finally, intrusion detection won't keep attackers out, but it will let you track them down. And setting up intrusion detection tools doesn't mean you have to fall for some proprietary vendor's story. See "Open-Source Intrusion-Detection Tools for Linux" for software that puts you in control.

The big news in security is the expiration of the RSA patent, which, along with the U.S. government's relaxation of export controls, means we can use open-source security tools such as OpenSSL everywhere. No more paying tribute to the RSA bandits to use their patent. That's great news, and we expect the Linux distributions to start integrating strong crypto everywhere.

With RSA and the government out of the way, what's stopping Linux from getting seriously secure? Nothing but us, any more. As *Linux Journal* moves into the post-excuses world of crypto everywhere, we'll be running regular articles on how to clean up your act, security-wise, and holding vendors to tighter standards of security cluefulness. We want our readers to learn about security from this magazine, not from a pager in the middle of the night.

Peace and Linux.

—Don Marti

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

[Advanced search](#)

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

[Advanced search](#)

RTcmix for Linux: Part 1

Dave Topper

Issue #78, October 2000

In the first part of this three-part series on real-time audio synthesis, we take you through the history and basis of RTcmix.

RTcmix is a software package that performs audio synthesis and effects processing. Basically, it's a real-time version of the Cmix language. Standard control over program execution is achieved via a C-style scripting language called MINC (MINC Is Not C). It has been open source long before the phrase was coined. That fact has been key to its evolution.

History

There have been literally dozens of RTcmix hackers over the past 20 years. I've done my best to give some credit as it relates to this article. You can also check the AUTHORS file included in the recent distribution (see "Getting and Installing" below).

Cmix was derived from the MIX program, a 20-track mixer written by Paul Lansky (at Princeton University) in 1978, which ran on VMS in FORTRAN on IBM mainframes. Synthesis abilities were added, and Paul ported Cmix to run on a PDP11/34 in 1983-84 under BSD2.9 UNIX. In 1985, he moved it over to Ultrix on a DEC MicroVAX. He and Lars Graf added the MINC parser in 1987, with help from Brad Garton and Dave Madole on various parts. Cmix lived happily under the NeXt (and other UNIX systems) for several years with its user base primarily in academic institutions. It has been and continues to be used both for instruction, computer music composition and research development.

Doug Scott and Paul Lansky ported an initial real-time version to SGI in 1995. While sometime in 1993, as a student and research assistant at Columbia University's Computer Music Center (CMC), I took a first stab at a Linux port which had to use Sox to unswap big-endian files. Boy was that fun. In late 1995, Brad Garton (CMC director) and I created a real-time version with a scheduler.

RTcmix was born, originally running under IRIX. Porting to Linux basically entailed swapping out the IRIX audio API with OSS (Open Sound System). The initial IRIX version was presented at the International Computer Music Conference (ICMC) '97. Shortly thereafter, Luke DuBois (also at the CMC) added an API to make writing TCP socket data easier. Doug Scott set up a mechanism to dynamically load instruments at run-time in 1997. John Gibson (at UVA) cleaned up the business of reading and writing different audio file formats in 1999 with an interface to Bill Schottstaedt's (at Stanford University) sndlib. John and I recently added two new abilities: routing audio via an internal "bus" mechanism and multichannel audio support. This work was presented this past year at the Society for Electroacoustic Music in the U.S. (SEAMUS) Y2K conference in Denton, Texas. Work also continues on controlling synthesis parameters in real time, creating various interfaces, and more robust socket support.

Setting up CVS (Concurrent Versions System) to help manage code has facilitated recent collaborations and development. The main core of RTcmix is released under the GPL (GNU Public License), with some different licenses for various components (e.g., instruments and effects) depending on their origin.

Getting and Installing RTcmix

In order to make use of RTcmix's real-time audio abilities, you need to install and set up the OSS audio drivers for Linux (see Resources). Support for ALSA (Advanced Linux Sound Architecture) remains untested but is planned. If you want to use RTcmix as a real-time audio processor, you need to make sure your audio card supports full-duplex operation. The Linux version of RTcmix can be downloaded via ftp from:

presto.music.virginia.edu/pub/RTcmix

The latest version as of this article is 3.0. The complete package exists in separate packages. RTcmix-3.0.0.tar.gz: the core engine and scheduler. insts.jg-3.0.0.tar.gz: instruments written by John Gibson. insts.mch-3.0.0.tar.gz: multichannel instruments. insts.std-3.0.0.tar.gz: standard suite of instruments. [At publication time, current versions of all of these packages is 3.0.2—ED]

I should probably comment on some terminology here. An RTcmix "instrument" can be either an effects processor (e.g., DELAY) or a synthesizer (e.g., FM synthesis).

Instrument packages should be unpacked into the same directory as the main code. So if you unpack the main code (RTcmix) to /usr/local/src, you'll end up with /usr/local/src/RTcmix-3.0.0. Untar instrument packages into that directory. In the case of insts.std.tgz, you'd end up with /usr/local/src/RTcmix-3.0.0/

insts.std-3.0.0. You can unpack the main code into any directory you like (e.g., your home directory or anyplace else).

In order to keep instruments and RTcmix versions consistent, we've added version tags to their directories. To make compilation simpler, you should symlink these to default names. For example, insts.std.tgz will untar into insts.std-3.0.0. You should symlink this as:

```
ln -s insts.std-3.0.0 insts.std
```

Once you've gotten everything unpacked and linked, you can set things up for compilation. There's no `./configure` utility here. You have to edit `makefile.conf` in the RTcmix root directory. Only a few lines need changing. Reading from the top of the file:

```
# Change stuff in here as needed

# The directory containing the RTcmix directory
TOPDIR = /usr/local/src

# The name of the RTcmix directory
CMIXDIR = $(TOPDIR)/RTcmix-3.0.0

# The dir that will contain links to the
# instrument dynamic shared objects
LIBDESTDIR = $(CMIXDIR)/shlib

# add new instruments to this list (e.g.,
# insts.std, insts.jg)
INST_DIRS = insts.base #insts.std insts.jg insts.mch
```

The only things you really need to potentially change are **TOPDIR**, depending on where you've unpacked RTcmix, and **INST_DIRS**, depending on which instrument packages you've downloaded.

Once you've made the appropriate changes, you can do a **make && make install** from the top-level RTcmix directory. This will compile everything and put a CMIX executable in `TOPDIR/bin` and shared libraries in `TOPDIR/shlib`. Make sure the bin directory is part of your **PATH**.

Making “Music”

The MINC scripting language provides both a powerful and easy-to-learn interface to RTcmix. More terminology: a MINC script is referred to as a scorefile. If you understand C, using MINC will be trivial. Some major differences are no semicolons at the end of each line and no functions (sorry). The art of using MINC (RTcmix) to make music is the primary focus of several semester-long college and graduate courses offered at several higher education institutions around the world. There's no way I can do it complete justice here. Instead, we'll take a look at some scorefiles which illustrate various features.

To execute a scorefile, simply pipe it into CMIX with stdin.

```
CMIX < scorefile
```

from the command line will fire it off. Internally, this syntax passes the scorefile to a parser (compiled by yacc/bison) which in turn passes events to a scheduler. When used in real-time performance mode (i.e., with external control sources), the RTcmix parser listens to a TCP socket, parsing data on the fly. In this instance, parser and scheduler are implemented using the linuxthreads library. The following scorefiles are included with version 3.0 of RTcmix and were designed by John Gibson.

Listing 1.

Listing 1 illustrates a reasonably simple scorefile which uses two RTcmix instruments. There are dozens more. WAVETABLE, as the name implies, is a wavetable lookup instrument. REVERBIT, also descriptive, runs a reverberation effect on an audio stream. In this example, the output of WAVETABLE is piped into REVERBIT, then out to your audio device and/or an audio file.

Let's step through the scorefile a bit:

```
rtsetparams(44100, 2)
```

This command is necessary for all RTcmix scorefiles. It sets up the audio device to run in stereo at 44100hz. An optional third parameter allows you to configure the audio buffer size to reduce latency in performance (e.g., when controlling RTcmix from another application). Default is 8,192 frames.

Next we declare which "instruments" we're going to use:

```
load("WAVETABLE")  
load("REVERBIT")
```

As mentioned above, RTcmix supports the dynamic loading of libraries. These commands load the instruments at run-time. This allows for a smaller memory footprint than loading all the instruments simultaneously. It also allows you to write a score that uses multiple instruments.

The next few lines are commented out. Uncommenting them will allow audio to be written to a soundfile as well as the audio device. The `set_option` command (de)activates various RTcmix features. For example: `set_option("AUDIO_OFF")` will turn off writing to the audio device—useful when your scorefile forces your computer to stretch beyond its means.

In order to map the audio from one source to another, eventually to your speakers or a soundfile, we call:

```
bus_config("WAVETABLE", "aux 0-1 out")
bus_config("REVERBIT", "aux 0-1 in", "out 0-1")
```

As mentioned earlier, one of RTcmix's newest features is the ability to route audio streams. In an attempt to keep tradition with established mixing convention, we've adopted a "bus" paradigm. Internal buses are labeled as "aux", which can be read from (as "in") or written to (as "out"). Input devices are labeled "in", and output buses (audio devices) "out." The convention for bus designation is: **bus_type** "number" [in or out for aux]. So "out 0" would be an output channel (e.g., a speaker) and "in 0" a line in or mic. This is not to be confused with "aux 0 in" which is the specification of an internal bus that is being read from by a particular instrument.

So, the statements above first route WAVETABLE's output to aux (internal bus) 0 and 1. Then REVERBIT's **bus_config** is set up to read from aux 0 and 1 (WAVETABLE's output) and write its output to a stereo audio device (out 0-1). It is possible to have multiple bus configurations for the same instrument. Each inherits the most recent call.

The next line in the score sets a global variable **totdur**. Variables are auto declared at init time (i.e., when the RTcmix parser reads your file). There are no reserved global variable names.

In order to set "control rate" update of internal synthesis parameters (e.g., amplitude envelope updates) we use the **reset** command. So **reset(2000)** updates 2,000 times every 44,100 samples—every 20.5 samples.

The following two lines are a bit less self-descriptive:

```
setline(0,0, 1,1, 5,0)
makegen(2, 10, 10000, 1, .5, .3, .1)
```

As with many packages that have developed over a number of years, RTcmix has its share of legacy issues. Unit generators in Cmix are defined by the **makegen** statement. Makegens are used for various internal aspects of a particular instrument: an amplitude envelope, waveform for a lookup, pitch vibrato, and so forth. The **setline** command is a related statement, used specifically to set an amplitude envelope. The time-value pairs set an amplitude of 0 at time 0, 1 at time 1 and zero at time 5. Time in this sense is relative and will be stretched accordingly to the duration of your "note" (see below). The WAVETABLE instrument needs some kind of waveform to read in order to make sound. It stores this information in "slot 2". Slot 1 is used for the amplitude envelope defined by the setline. Yes, you could substitute a makegen command

for that setline (gen24). Gen10 specifies loading a sinusoid into the respective slot (in this case 2). The size (in samples) of the wavetable in this case is 10,000. Remaining arguments are amplitudes for successive harmonics. That's about as complicated as it gets, at least in this article.

After assigning some more global variables, we seed the random number generator with:

```
srand(3284)
```

The actual music making is done by the following lines:

```
for (st = 0; st < totdur; st = st + .45)  
    WAVETABLE(st, dur, amp, freq, random())
```

This is a simple for loop which executes until st is greater than totdur (defined earlier). The parameters to WAVETABLE (as well as all other RTcmix functions and instruments) are referred to as p-fields. In the case of WAVETABLE:

- p0 = start time
- p1 = duration
- p2 = digital amplitude (in this case between 0 and 32767)
- p3 = frequency (in hz)
- p4 = stereo spread

For p4 = 0 100% of the output will go to channel 0 (defined by bus_config above) and pf4 = 1 sends it all to channel 1. A value of 0.5 splits between both channels evenly. Note that the random() function call returns a floating point value between 0 and 1u.

The last few lines of the score reset the amplitude envelope with **setline(0,1,1,1)**, because we want a slightly different one for our REVERBIT function, which we call with: **REVERBIT(st=0, insk=0, totdur, amp, revtime, revpct, rtchandel, cf)**.

The p-fields for REVERBIT are:

- p0 = start time
- p1 = inskip, which is the time to start reading input. (Use 0 for bus_configs; some other value when reading in from a file.)
- p2 = duration
- p3 = amplitude multiplier (generally between 0 and 1)
- p4 = reverb time
- p5 = reverb percentage
- p6 = right channel delay

- p7 = cutoff frequency for low-pass filter (in hz)
- p8 (optional) = apply DC blocking filter (boolean)

That's it. Simply type the following to execute the score:

```
CMIX < WAVETABLE_REVERBIT_1.sco
```

Listing 2 is a musically simpler example, but illustrates the powerful ability of RTcmix to process an incoming audio stream in real time.

Listing 2.

To set up our audio device for simultaneous reading and writing, we use the `set_option` command again:

```
set_option("full_duplex_on")
```

It's necessary to have this command executed before the `rtsetparams` call, as it sets an internal flag that `rtsetparams` uses when performing actual setup of the audio device. As before, we set up the device for 44K stereo, but this time decreasing the internal buffer size for real-time performance to 512 frames.

To tell RTcmix to read from the audio device, we use:

```
rtinput("AUDIO", "MIC")
```

The **MIC** argument is necessary only in IRIX versions of RTcmix. We could just as well say:

```
rtinput("some_file.aiff")
```

to read an audio file instead.

Conclusion

RTcmix is an incredibly powerful and flexible package because of the work of many people, and possible because of its open-source nature. Different features have been added by different people over the years. It continues to evolve. The Linux operating system has been an excellent platform both for development and performance. With the recent advent of high-end, multichannel digital audio card support for Linux (e.g., the RME Digi96 series), the abilities of RTcmix grow in conjunction with the OS.

This article only scratches the surface of RTcmix's potential. Future articles will discuss in greater depth the process of writing RTcmix instruments and controlling them in real time (e.g., with nice open-source GUI packages like GTK).

In the end, it remains amazing to think that a computer, operating system and software package can work together well enough to make music. It is, however, by no means new thinking. Ada Lovelace apparently debated at length with Charles Babbage the virtues of this “new computing device”. His contentions had to do with an unbeatable chess player, hers about a device that could compose and create music of any type or degree of complexity.

Resources

email: topper@virginia.edu

David Topper (topper@virginia.edu) is the technical director for the Virginia Center for Computer Music at the University of Virginia. Linux has been his primary OS since downloading 40 floppies' worth of Slackware (kernel 1.0.9) as a CS undergrad. It is one of his firmest beliefs that the computer can be to the human mind and spirit what the telescope was to the ancient astronomers, provided free software like Linux continues to thrive. His web page is at www.people.virginia.edu/~djt7p

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

[Advanced search](#)

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

[Advanced search](#)

Apache User Authentication

Ibrahim F. Haddad

Issue #78, October 2000

A guide to setting up user authentication for the Apache web server running on Linux, using the plaintext file method.



[The Apache Web Server](#)

Apache has been the most popular web server on the Internet since April of 1996. The Netcraft February 2000 Web Server Survey found that over 58% of the web sites on the Internet are using Apache, thus making it more widely used than all other web servers combined. This success can be attributed to the fact that Apache provides a robust and commercial-grade reference implementation of the Hypertext Transfer Protocol (HTTP). Also, it is reliable, configurable, highly scalable, well documented, open source and free.

Getting Started

First, you must have Apache and the **htpasswd** utility installed on your Linux machine. If you do not, you can download the latest copy of Apache (version 1.3.12 is available now) from the Apache Organization web site and install it on your machine. htpasswd comes with Apache.

The Authentication Process

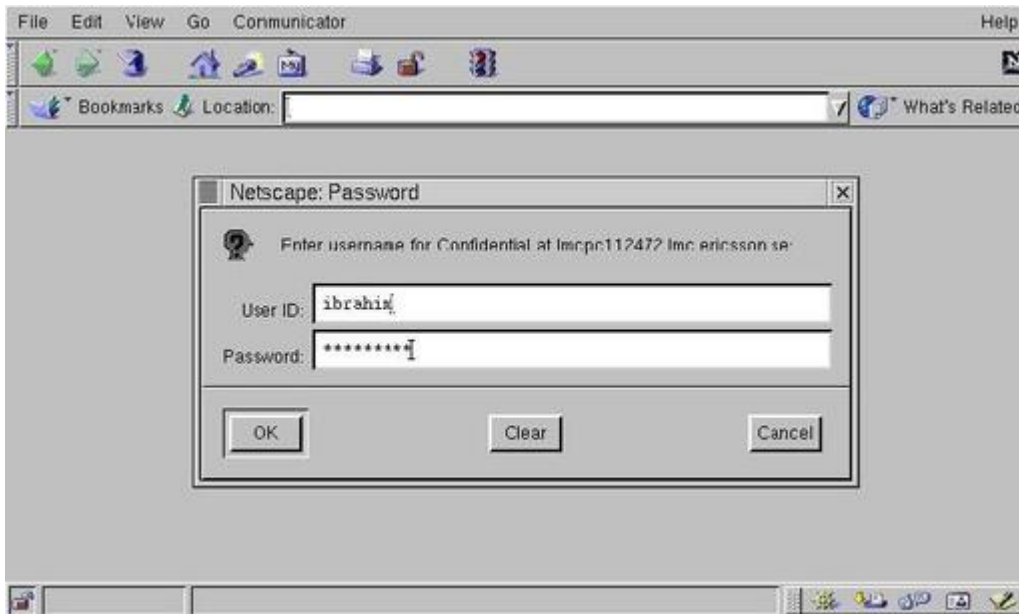


Figure 1. Sending User Credentials

Authentication is a simple yet very important principle—the client sends its name and password to the server (see Figure 1). Apache checks if the credentials are valid, and if so, returns the requested page (see Figure 2).

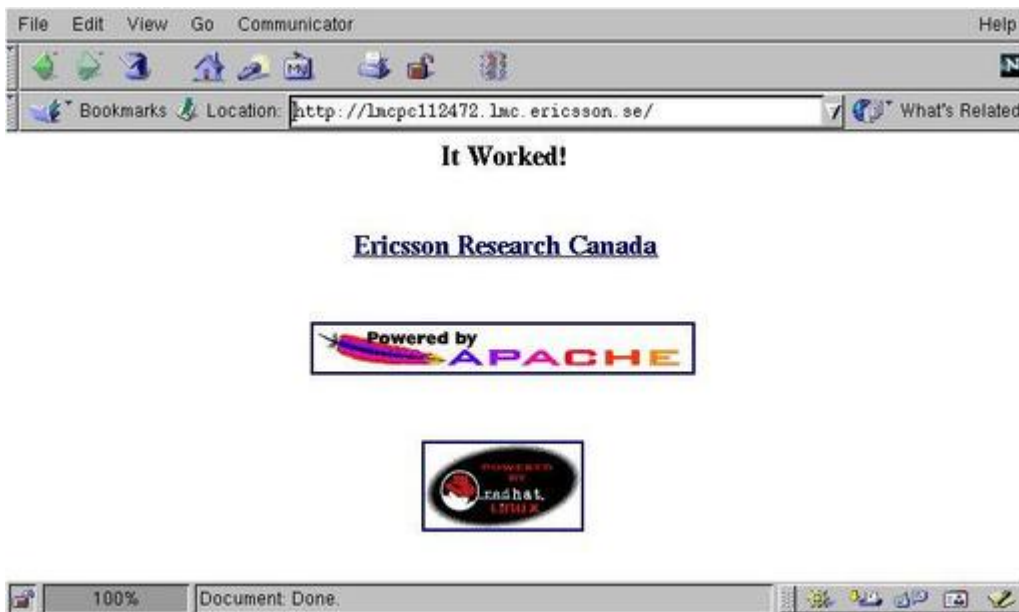


Figure 2. Access is Granted

If the user is not allowed to access the page or the supplied password is not valid, Apache returns a 401 status (i.e., unauthorized access). The browser will then ask the user to retry their user name and password (see Figure 3).



Figure 3. Failed Access

Restricting Access Methods

Restricting access to documents is usually done based on either the host name of the browser or user credentials (user name and password). The decision to adopt either method depends on your environment settings. For example, if you want to restrict documents within a limited environment such as a department, you can use the client host name. Otherwise, if you want to grant access on an individual basis, or if the people who are allowed access are dispersed, you would then require a user name and a password.

Restricting Access by User Name and Password

In order to set user authentication using a user name and a password, we need to follow two steps. First, we create a file containing the user names and passwords. Then, we inform the server which resources are to be protected and which users are allowed to access them.

The first step towards configuring authentication is to set up a list of users and their corresponding passwords. This list is saved in a file called, for example, users. This file should not be saved under the root directory for security reasons. Therefore, for my setup, I saved it under /etc/httpd. For an organized setting, you can create a directory under /etc/httpd, perhaps called users-files, and save the file in it. The file format is similar to the standard UNIX password file, consisting of a list of user names and an encrypted password for each, separated by a colon.

The htpasswd Utility Mini-HOWTO

htpasswd is a utility supplied with the Apache package that allows us to create a user's file containing user passwords in order to add or modify them. On my machine, powered by Red Hat 6.1, Apache 1.3.9 and htpasswd were installed as

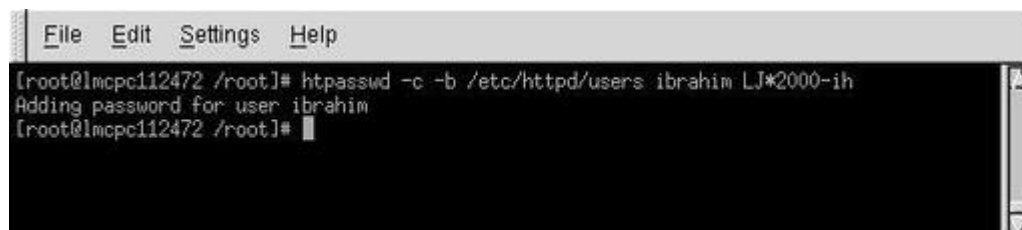
part of the standard installation. To see how htpasswd works, type htpasswd at the shell prompt and the following messages will be returned:

```
Usage:
htpasswd [-cmdps] passwordfile username<\n>
htpasswd -b[cmdps] passwordfile username password
-c Create a new file.<\n>
-m Force MD5 encryption of the password.
-d Force CRYPT encryption of the password (default).
-p Do not encrypt the password (plaintext).
-s Force SHA encryption of the password.
-b Use the password from the command line rather than prompting for it.
On Windows and TPF systems the '-m' flag is used by default.
On all other systems, the '-p' flag will probably not work.
```

According to the usage statement, to create a new users file and add the user ibrahim to the file /etc/httpd/users, we type the following command:

```
htpasswd -c -b /etc/httpd/users ibrahim LJ*2000-ih
```

The **-c** flag is only used the first time we use htpasswd to create a new users file. When we run htpasswd with the **-b** flag (please see the Security Issues section), we will not be prompted to enter a password for ibrahim since it was passed from the command line. Other users can be added to the existing file in the same way, except that the **-c** argument is not needed since the file already exists. If this option is used when adding other users, the file is over written and the old users are lost.



```
File Edit Settings Help
[root@lmcpcl12472 /root]# htpasswd -c -b /etc/httpd/users ibrahim LJ*2000-ih
Adding password for user ibrahim
[root@lmcpcl12472 /root]#
```

Sample User Password File

After I created a “web account” for myself, I added a few other accounts and my /etc/httpd/users file looked like:

```
ibrahim:40gvm/LYXsk4a
chady:XygEnj0pSDx9A
julie:3zwRHLJDrR/9s
carla:IFPYPtrekJLxE
karine:jem9XkbaLFaTA
```

Configuring the Server

Now that the password file has been created, the server must be notified that access will be restricted based on the user names and passwords found in this file (/etc/httpd/users). This protection method provides access control to individual files as well as to directories and their subdirectories. The directives to create the protected area can be placed either in a .htaccess file in the

directory to be protected, or in a <Directory> section in the access.conf file (located in /etc/httpd/conf).

Restricting Directories Using .htaccess

To restrict a directory from within a .htaccess file, first ensure that the access.conf file allows user authentication to be set up in a .htaccess file. This is controlled by the **AuthConfig** override. The access.conf file should include **AllowOverride AuthConfig** to allow the authentication directives to be contained in a .htaccess file. Now, to restrict a directory to any user listed in the password file, create a .htaccess file (inside the directory to be protected) containing the following directives:

```
AuthName "Confidential"  
AuthType Basic  
AuthUserFile /etc/httpd/users  
require valid-user
```

.htaccess Contents Description

The **AuthName** directive specifies a realm name for this protection. Once a user has entered a valid user name and password, any other resources within the realm name can be accessed with the same user name and password. This can be used to create several areas that share the same user name and password.

The **AuthType** directive tells the server which protocol is to be used for authentication. Another method, **Digest**, offering higher security features, is also available in the **mod_digest** module. Using MD5 Digest authentication is quite simple. Set up authentication using **AuthType Digest** and an **AuthDigestFile** instead of the normal **AuthType Basic** and **AuthUserFile**. Everything else should remain the same.

The **AuthUserFile** directive tells the server the location of the user file created by htpasswd. Similarly, when using a group file (described in the next section), use the **AuthGroupFile** directive to tell the server the location of a group's file.

Last, the **require** directive tells the server which user names from the file are valid for particular access methods. In this example, the argument **valid-user** tells the server that any user name in the password file can be used. However, we can configure it to allow only specific users. An example of this would be:

```
require ibrahim julie carla
```

This directive allows only ibrahim, julie and carla to access the documents after they enter a correct password. If any other user tries to access this directory,

even with the correct password, they would be denied. This approach is very useful for two reasons:

- Different areas of the server can be restricted to different people with the same password file.
- If a user is allowed to access different areas, he has to remember only a single password.

Using a Group File

In some cases, we want to allow only selected users to access a particular directory. A lazy way of doing this is by listing all the allowed user names on the **require** line. This is not encouraged since, if there are many users, the file will become quite large. Luckily, there is a nice way around this problem—using a group file which operates in a similar way to standard UNIX groups. Any particular user can be a member of any number of groups. We can then use the **require** directive line to restrict users to one or more particular groups. For example, we could create a group called `research-staff` containing users who are allowed to access the research department pages. To restrict access to just users in the `research-staff` group, we would use this directive:

```
require group research-staff
```

Multiple groups can also be listed, and **require user** can also be given, in which case any user in any of the listed groups, or any user listed explicitly, can access the resource. For example:

```
require group research-staff admin-staff  
require user julia carla
```

allows any user in group `research-staff` or group `admin-staff`, or both users `julia` and `carla`, to access this resource after entering a valid password.

A group file consists of lines giving a group name followed by a space-separated list of users in that group. For example:

```
research-staff:chady karine  
admin-staff:ibrahim julie
```

The **AuthGroupFile** directive is used to tell the server the location of the group file. The catch with using a group file is that the maximum line length within the group file is 8KB. However, to get around this limit, we can have more than one line with the same group name within the file.

Potential Slowdowns

Using a plaintext file to maintain user names and passwords is easy and straightforward. Nevertheless, employing this method with a large number of users causes a lot of processing at the server side to search the file for the credentials in question; this adds to the server load. Moreover, processing has to be done for every request inside the protected area; even though the user only enters their password once, the server has to re-authenticate them on every request due to the stateless nature of HTTP. Therefore, the server does not remember any information about a request once it has finished and must resend the user name and password on each request.

Faster Access

Much faster access is possible using DBM format files. This allows the server to do a very quick lookup of names, without having to read through a large text file. The slight drawback of this method is the complexity of managing DBM files as compared to managing plaintext files. There are various add-on modules which allow user information to be stored in databases. Aside from the DBM format (**mod_auth_dbm**), user and group lists can be stored in DB format files (**mod_auth_db**). Full databases can also be used such as mSQL (**mod_auth_msql**), Postgres95 (**mod_auth_pg95**) or any DBI-compatible database (**mod_auth_dbi**).

Security Issues

There are a couple of security considerations regarding the password files managed by `htpasswd`. First, files containing users' information such as `/etc/httpd/users`, should be outside the web space of the server—they must not be fetchable by a browser. Secondly, the use of the `-b` flag with `htpasswd` as shown in Figure 4, is discouraged since when used, the unencrypted password appears on the screen.

Conclusion

Authentication is vital and necessary for most web servers. Apache has proven its reliability, and has an excellent record of stable performance and trustworthy security. Using Apache's authentication features, we can combine a cost-effective way to secure our documents using the most popular web sever running on Linux.

Resources



email: ibrahim.haddad@lmc.ericsson.se

Ibrahim F. Haddad (ibrahim.haddad@lmc.ericsson.se) is a senior member of technical staff at Ericsson Research Canada based in Montréal. He researches distributed-object technologies and web servers performance at Concordia University as a D.Sc. Candidate. Ibrahim would like to take this opportunity to thank his parents for all their help and support, not to mention the countless sacrifices, in the last twenty-five years.

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

[Advanced search](#)

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

[Advanced search](#)

Distance Education Using Linux and the MBone

Kelly Davis,

Tom Miller

Charles Price

Issue #78, October 2000

There is more to the Internet than sending JPGs. See how Linux and the MBone addresses the needs of distance learning.

One of the most promising applications of the Internet is distance education. The most significant advantage of Internet connectivity over previous communication paradigms for distance education is that a high degree of interaction among teachers and learners is possible. Education is a highly interactive process. If it weren't, schools wouldn't be needed—textbooks alone would be sufficient. The fact that the Internet protocols support two-way communication and virtually unlimited possibilities for integrated multimedia (given sufficient bandwidth and quality of service) presents unprecedented opportunities for extending the reach of education beyond the traditional classroom.

Most applications of Internet technology to distance education have been web-based courses. In a web-based course, the course content is developed as modules in HTML, and interaction takes place asynchronously through discussion boards or plain old e-mail. Some web-based courses also support synchronous interaction through chat sessions. Some advantages of web-based courses are that they work well at modem bandwidths and require only a web browser on the student's computer, thus maximizing the possible audience and minimizing the technical support requirements. However, a quality web-based course requires a great deal of effort to produce; and in disciplines such as engineering where the content must be updated frequently, maintenance of the course can also be significant. Another important limitation is that interactions are essentially limited to textual media. In answering a question for a student in an engineering class, the teacher often needs to construct or mark

up a diagram while explaining the concept. This is best accomplished in an environment which supports real-time audiovisual interaction, which is why the traditional face-to-face classroom has survived the last few hundred years so well.

With this in mind, the “MBone virtual classroom” was developed at North Carolina State University to allow students to attend live engineering classes from a remote location by “tuning in” to the class from a workstation. The concept was to replicate as nearly as possible the face-to-face environment with real-time interaction including audio, video and graphics. To solve the problem of having many remote students attending the classroom virtually, IP-multicast and the MBone tools were employed. The virtual classroom has been in use since the fall of 1996 to provide access to engineering classes to students at several locations in North Carolina. This article describes how IP-multicast and the MBone tools were used to create the virtual classroom environment, and the development of DETA, the Distance Education Teaching Assistant, a Tcl/Tk-based wrapper which provides a simple, unified interface to the many hardware and software components of the virtual classroom.

IP-Multicast and the MBone

IP-Multicast, the class-D addressing scheme in IP which makes the MBone network possible, was developed by Steve Deering in his PhD thesis at Stanford, and later developed and implemented at Xerox Palo Alto Research Center (PARC). The first multicast tunnel was established between BBN and Stanford University in the summer of 1988. The Internet Multicast Backbone (MBone) was subsequently established as a virtual network of multicast tunnels over the existing Internet. In 1992, the same year that the Internet grew to one million hosts and Mosaic was created at NCSA, the MBone carried its first real-time audio and video traffic.

IP multicast is useful in that it provides an efficient mechanism for “broadcasting” data over the Internet. It is best understood in comparison to IP unicast. When sending the same data to multiple clients using unicast, multiple separate connections to those clients must be opened. When the number of clients grows significantly, the load on the sender increases dramatically. With IP multicast, the data needs to be sent only once. The multicast-enabled network sends copies of the data to all of the clients who wish to receive the data. In this way, the sender transmits the data only once, regardless of the number of clients wishing to receive the data. It is very similar to a television broadcast, where a single transmitter sends out a single video transmission, and anybody within range of the signal can receive the transmission. Because it is more efficient at sending the same data to multiple recipients, multicast is ideal for multimedia network applications such as video-conferencing or live netcasts.

The ability to send and receive IP multicast is primarily dependent on your network. The network's routers must know how to deal with multicast packets. A few years ago, there existed very few routers capable of handling multicast packets. At that time, a method was needed to send multicast packets over networks designed to handle only unicast packets. This method became the Virtual Multicast Backbone, or MBone. Software that uses the MBone essentially packages multicast packets inside of unicast packets, which non-multicast-enabled routers know how to handle. Multicast-enabled routers are able to identify and process the multicast packets, as well as computers running MBone software.

The MBone Tools

The primary components of the distance education software used at NC State are the MBone tools. These are **vic**, **rat**, **wbd**, and **sdr** are available for download in binary and source form at University College London's (UCL) web site at <http://www-mice.cs.ucl.ac.uk/multimedia/software/>. **vic** is an MBone video-conferencing tool. It was originally developed by Steve McCanne and Van Jacobson at the Lawrence Berkeley National Laboratory (LBNL) Network Research Group. The version being used at NC State is currently under development at UCL. This version provides video-capture support using Video4Linux, so many existing video-capture cards are compatible with it. It incorporates a number of codecs including H.261 and H.263. It provides controls to adjust frame rate, bandwidth, and video quality, as well as many other options. Users can switch between thumbnail and full-screen video windows, and switch between a number of video formats. **vic** runs in multicast-conference mode or point-to-point-unicast mode.

The Robust Audio Tool (**rat**) is an MBone audio-conferencing tool. **rat** was developed by UCL's Networked Multimedia Research Group. There are two versions of RAT: a stable, toll-quality (i.e., telephone-quality) version 3, and an improved, though experimental, high-quality version 4. **rat** supports both ALSA (Advanced Linux Sound Architecture) and OSS (Open Sound System), so it is compatible with a large number of sound cards. **rat** provides a number of audio codecs, as well as packet-loss concealment schemes. Other features include automatic gain control, silence suppression and encryption. **rat** also provides a graphical interface showing conference participants and audio levels. Like **vic**, **rat** can operate in point-to-point-unicast mode or in multicast-conference mode.

wbd is an MBone shared whiteboard. It allows a number of participants in a conference to share a single whiteboard workspace. It was originally written by Julian Highfield at Loughborough University. The most recent development work on it has been done by Kristian Hasler at UCL. **wbd** is compatible with the original LBNL whiteboard, **wb**, developed by Steve McCanne. Because **wb** is

available only in binary form for UNIX platforms, Julian Highfield wrote **wbd** primarily to fill the need for a Windows version of **wb**. Since the source is freely available, we have chosen to use **wbd** over the binary-only **wb** on Linux. **wbd** has a standard set of whiteboard features, such as font, color and line-width options, text input capability, drawing tools and various page orientations. **wbd** can import both PostScript and text files. **wbd** was designed to work in both point-to-point and shared multicast modes, though currently only the multicast mode functions properly in Linux.

Instead of each user in a conference connecting to every other user, MBone users join a multicast group. Anything sent to the group is received by all current members of the group. None of the MBone tools discussed so far provide any means of locating or advertising these groups. This is accomplished through the session directory tool, **sdr**. In a sense, the session directory is like a television guide which shows all the currently available "shows" on the MBone. **sdr** was originally written at UCL and was modeled after another LBNL tool called **sd**. When the user loads up **sdr**, a listing of all public and private MBone sessions appears. To get more information about a specific session, the user clicks the session name. To join a session the user clicks the join button for that session and **sdr** then loads up the various tools needed to participate in the session. To create a new session, the user clicks the New button and enters various information about the session. **sdr** then generates the multicast addresses and advertises the session for other users to see.

The bandwidth requirements for the MBone tools are relatively low by current standards. Each video source requires only about 128KBps at a rate of ten frames per second. The audio requires about 64KBps at telephone quality. Higher frame-rate video is possible, but we've found that high-quality whiteboard data in combination with good-quality audio more than compensate for the slow video. The video mainly orients the participant and provides visual cues, while the actual content is provided via the audio and whiteboard data. For some classes, it is more important to provide full-motion video, and when adjusted appropriately, **vic** can provide this.

DETA, the Distance Education Teaching Assistant

The MBone tools were not specifically designed with distance education in mind. In fact, though interoperability and uniformity between the applications was a goal of the original designers, it had yet to be achieved when the first MBone classes began at NC State. Additionally, the tools themselves have a relatively steep learning curve for nontechnical users. For example, in order to create a new session, a number of technical options have to be set in **sdr**, including Type of Session, Session Scope, Type of Media and Session Time. Experienced users have little difficulty navigating these options, but to meet the needs of distance education, users have to be able to operate the tools with

only minimal training. In order to accomplish this goal, the distance education Teaching Assistant was created. DETA is a wizard-style interface to the MBone tools. It hides all of the technical options the inexperienced user should not have to deal with. Originally, it primarily functioned as a simplified interface for starting Mbone-based Distance Education classes. Over the years, a number of features have been added, including the ability to give preprepared presentations and automatic digital archiving of classes.

The underlying purpose behind DETA is to meld the various tools and supporting programs into a single, easy-to-use application. DETA is written in the Tcl/Tk interpreted scripting language. The primary reason for this choice has to do with the MBone tools. All of them are Tcl-based applications written in C or C++. This not only provides great flexibility within the applications themselves, but also allows for basic interoperability between them. This interoperability is achieved through Tcl's Send command. Send allows one Tcl interpreter to execute commands remotely within another applications interpreter. In DETA, the Send command is primarily used to remotely operate SDR, as well as to set options within the other MBone tools.

One of the drawbacks of using the Tcl Send command for communication is that it doesn't hide any of the various tool's internal implementation. This becomes an issue when that implementation is changed, or a new tool is added. The core DETA script then has to be altered to adapt to the changes in the tools. One solution to this problem is to modify the various tools directly so that they communicate via a common, implementation-independent bus. This would require separate modified versions of the MBone tools. Rather than use modified versions of the MBone tools, it was decided to modularize the code that communicates with the various tools into separate interface packages. The interfaces contain all of the tool-specific Tcl code in string arrays. These arrays are sourced by the primary DETA script at startup. In this way, the core script can be modified without affecting implementation-dependent code, and new or updated tools can be added simply by creating new interface modules.



Figure 1. Mode Selection Screen for DETA

When the user first starts DETA, he or she is given four mode options, as seen in Figure 1. The first is to create a new session. This creates a new multicast session that will be announced while the session is active, and which will expire after the session becomes inactive. By creating the session, the user becomes the designated host of that session. The second option is to host an existing session. This mode is included so that users can host permanently existing sessions. At NC State, a session directory server was written by Troy Holder that continuously announces class sessions. In this way, semester-long class announcements do not have to be re-created at every meeting. The third option is to join an existing session. This is the option that a remote participant would select. It allows the user to select a desired session from a list of available sessions and then join it. The last option is to play a recorded session. This mode is used to select a previously archived session for playback via a remote VCR server. Once the mode is selected, the user enters information about the session. Figure 2 shows the information required when creating a new session. This includes the name of the session, the lecturer name, and the MBone tools to use in the session. This is saved between sessions and usually doesn't need to be re-entered. Once the options for the session have been set, DETA creates the session and loads the MBone tools.

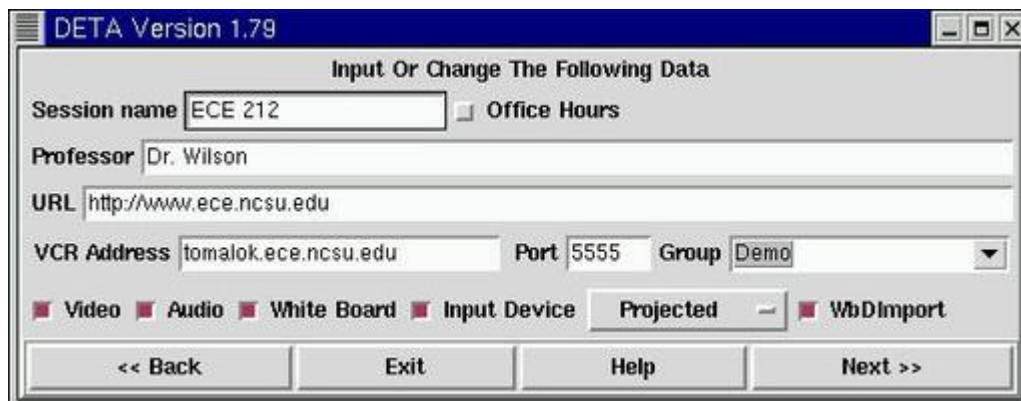


Figure 2. User Information Screen for DETA

In addition to melding the core MBone tools, a number of features have been added to DETA. The first is a tool called WBDImport developed at NC State. This tool is modeled after WBImport, which was written by Van Jacobson. It allows the user to give a PowerPoint-style presentation using the shared whiteboard tool. The format used for slide files is generic PostScript. The user indicates which files will be used at startup, either by specifying a directory containing the slides, or by specifying a text file containing a list of all the files and their locations. During the session, a WBDImport window lists all of the slides specified at startup. To show a slide, the user simply clicks the name of the desired slide and it is loaded into the shared whiteboard. This tool allows professors to prepare a set of slides in advance and then annotate them during a lecture. The slides and the annotations are sent out live to all the participants

in the session. Using preprepared slides in this way has been found superior to handwriting for classes requiring large amounts of written notes.

The audio tool used in DETA allows only one user to speak at a time. If there is a question at a remote site, it is important that there be some way to signal the host so that the question can be asked. A tool called the Electronic Hand Raiser provides this capability. Remote users have a control panel with a button labeled "Ask Question". When they click this, the session host will hear a tone and see a message button indicating who the question is from. To acknowledge the question, the host clicks the message, and then allows the remote site to ask their question. There is also a "Cancel Question" button at the remote site if the remote user wishes to withdraw their question. In addition to dealing with the problem of half-duplex audio, this tool provides a form of floor control that matches traditional classroom protocols (See Figure 4). Figure 3 shows a live DETA session with the Mbone video, audio and whiteboard tools, as well as the Electronic Hand Raiser and WBDImport tool.

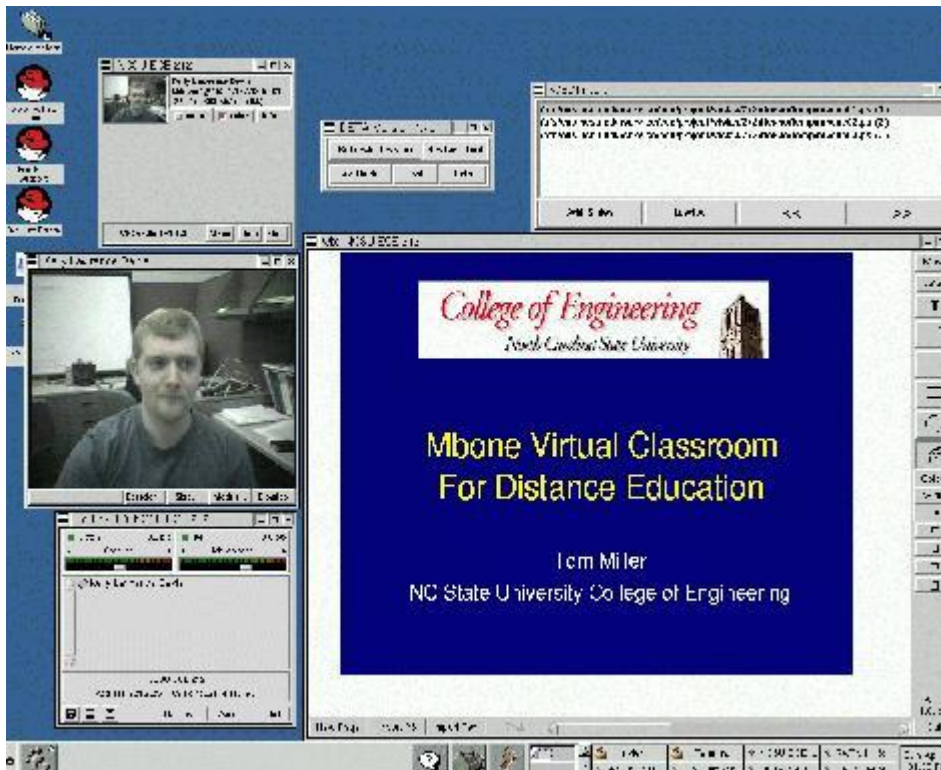


Figure 3. DETA in "host" Mode

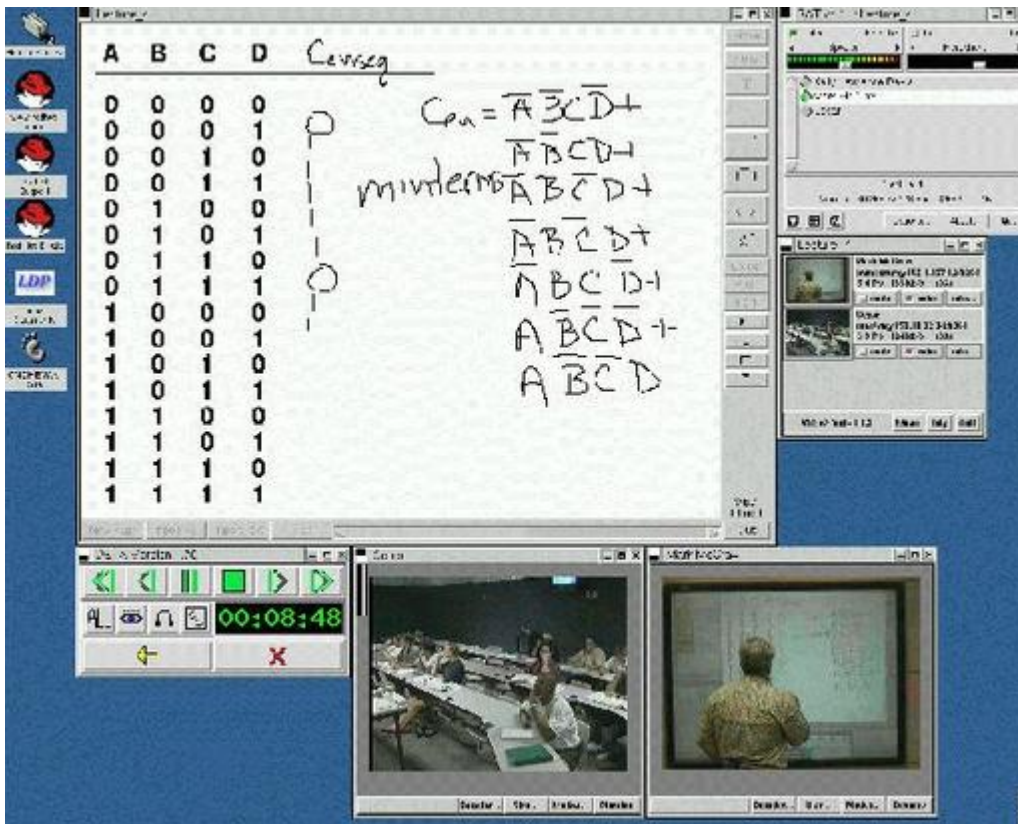


Figure 4. DETA Playing Back a Session

Another feature of DETA is integrated recording and playback of sessions. To record a session, the host checks the record option and the session is automatically recorded on a separate VCR server. To play back a session, the user selects the play session option and then chooses the desired session. Once the session is selected, DETA provides a VCR-style interface (see Figure 4). This interface also allows the user to start the various tools and provides standard VCR controls such as play and rewind. The VCR server itself is a separate Tcl script that DETA communicates with via TCP/IP sockets. The primary function of the server is to start and stop a Java application called mVCR. mVCR was written by Peter Parnes at the Lulea University of Technology in Sweden. It is capable of capturing and playing back multicast data streams for the Mbone audio, video and whiteboard tools. The DETA VCR Server and mVCR, in combination with DETA, provide immediate on-demand playback of sessions. As soon as a class session is complete, that session is available for remote playback. This greatly reduces the cost and overhead necessary to provide time-delayed classes to remote sites. Additionally, students who missed a class or wish to review a class can individually replay recorded sessions.

Issues with Recording and Playback

The recording and playback of DETA sessions is not entirely without problems. The recording system works by capturing all the network packets received at a client station during the initial live session, encoding them along with a time

stamp and writing them to disk. On playback the time stamps are used to deliver an “identical” stream of network packets that will exactly mimic the original session and preserve the synchronization of the audio, video and whiteboard. The quality of the recording (or the live session) can be affected by packet loss due to network congestion, processor loading, or other sources. Generally speaking, the loss of a single packet does only minimal damage to the audio or video recording. A missed syllable during playback is easily filled in from the context, and a missed video frame causes only a momentary pause or distortion. This is often not the case for the whiteboard. One of the ways faculty use the whiteboard is to load preprepared PostScript slides containing figures or equations, and then use the mark-up capability of the whiteboard to provide additional information and lead the class through the discussion of the presented material. A typical PostScript image, even when compressed, may be 10 to 50KB. In order to stream this to the client systems, the image is broken into multiple network packets by the server and reassembled at the client prior to rendering. If even a single packet is lost in transit, the reassembled image will be incomplete and the rendering will fail. The entire image will be absent from the client whiteboard. The whiteboard protocol contains a feature for dealing with this problem during a live session. Each network packet is numbered so every client can identify when a packet is missing. That client can then send a request to all the other session participants asking for the packet to be retransmitted, and any client that has the requested packet can respond. This late delivery mechanism works well for the whiteboard because it is less sensitive to the synchronization. Unfortunately, the mVCR recording tool is passive. It simply records the packets that are received without requesting retransmission of missing packets. Whenever a PostScript image is received with any packets missing, it will be incomplete in the recording, and will not appear during playback. This can significantly degrade the quality of the learning experience, since the students viewing the playback are unable to see the prepared material.

To overcome this difficulty, a repair tool was developed: **fix_wb_recording.pl**. The Perl tool simply parses through the whiteboard recording, locates any PostScript images with missing data, and replaces them with complete images. Parsing the recording requires detailed knowledge of both the mVCR recording format and the wb data format. Neither of these is well documented but sufficient information is available on the Web. The mVCR recording format is described by Peter Parnes at www.cdt.luth.se/~peppar/progs/mMOD/doc/fileformat.txt. The mVCR format is just a header containing basic setup information (multicast address and port number) and any number of wrapped wb data packets. The wb data format has never been published but Lars Rasmusson has posted a reverse-engineered description of the format at www.it.kth.se/~d90-lra/wb-proto.html. While it is known that this description is incomplete and incorrect in some details, it was sufficiently accurate for the

purpose of repairing the mVCR recordings. Within the wb data format, PostScript images appear as a sequence (not necessarily consecutive) of draw commands that encode the PostScript data and issue the command to the client to render the complete image.

Whenever the parsing routine encounters a PostScript image in the recording, it opens each PostScript file that was used in the original session and searches for a matching data block. Since most instructors prepare all of their PostScript images using a single software tool, the images often contain large sections that are identical. Therefore it is usually necessary to check several recorded data blocks before one is found that produces a unique match with one of the original images. Once a unique match is found, each draw command associated with the recorded image is deleted and new draw commands are inserted containing all of the data blocks of the complete image and an associated render image command. After a recording has been completed, DETA gives the user the option of post-processing the slides with the repair tool. If this option is chosen, DETA sends the actual PostScript slides to the VCR server and runs the `fix_wb_recording.pl` tool. Once this was implemented, problems with the quality of the recorded sessions dropped to nearly zero.

MBone Classes at NC State



Figure 5. The MBone Classroom at NC State

North Carolina State University has been running MBone classes since the fall of 1996. In this time, a number of undergraduate engineering courses have been delivered to participating universities and community colleges. At NC State, a classroom was constructed specifically for distance education (see Figure 5). This classroom contains seating for local students. There are two remote-control video cameras, one for the lecturer and one for the students. There are three large television monitors on which the computer screen is shown to the local students. There is a control area with a computer

workstation, document camera, two small TV monitors and an AMX controller. The AMX controller controls the cameras as well as the other audio and video sources. It also provides a central control to operate the other equipment in the room. The most innovative feature of the classroom is a digital projection SMARTBoard mounted on the front wall of the classroom. The SMARTBoard is an input device manufactured by SMART Technologies. It is essentially a touch-sensitive version of a standard office whiteboard. Whatever is written on the surface of the SMARTBoard is transmitted to the computer. The projection version used in the classroom works in combination with an LCD projector to project an image of the computer screen directly onto the SMARTBoard. In this way, the lecturer essentially writes directly on the computer screen. The SMARTBoard is large enough that the local students see directly what the lecturer is writing on the SMARTBoard. This technology provides a natural closed-loop interface to the computer that closely parallels the traditional classroom blackboard.

Generally, a teaching assistant operates the computer equipment and cameras while the instructor lectures. This frees the instructor from having to deal with any technical issues while the class is in session. One of these issues relates to the floor-control aspect of DETA. The floor-control provided by the Electronic Hand Raiser is purely voluntary, and requires all participants to follow the correct protocol. Often we have found that remote participants will scribble onto the shared whiteboard to get the lecturer's attention, or the instructor will simply fail to acknowledge incoming questions. A good solution to the floor-control issue remains to be found and usually it is the assistant's responsibility to help the lecturer acknowledge any questions. Another associated issue occurs when a local student asks a question. The lecturer must repeat the question in order for it to be heard by the remote sites. Often the lecturer will mysteriously stop speaking for a moment, and then start answering a question that none of the remote participants ever heard asked. Remote participants are then forced to either figure out what had been asked, or interrupt the flow of class and ask the lecturer what the original question was. One solution to this problem has been to give students their own microphones. Unfortunately, this relies either on them remembering to activate the microphone, or on distracting continuous presence audio.

The overall response by students to the MBone classes has been positive. The interactive capabilities provided by the MBone tools are far superior to videotapes or broadcasts. They provide a richer educational experience, more similar to a traditional classroom. Most instructors have been able to adapt well to the technology, though there exists somewhat of a learning curve for those accustomed to traditional classroom teaching. One of our primary aims in the future is to flatten this learning curve and make the technology more transparent to the user. Ideally, an instructor should be able to walk into a

classroom, activate the equipment, and begin lecturing immediately, without giving any more thought to the underlying technology. While this goal has yet to be achieved, we feel that the MBone tools and DETA, in combination with the Linux platform, represent a highly usable and cost-effective vehicle for the delivery of interactive distance education. For more information, as well as links to all of the DETA and MBone software, visit our web site at <http://www.engr.ncsu.edu/deta/>.

Acknowledgements



Kelly Davis (kldavis4@eos.ncsu.edu) is a graduate student in the Department of Electrical and Computer Engineering at NC State University and is currently pursuing his PhD. He began using Linux while an undergraduate at the University of Central Florida in 1995. He enjoys computers, outdoor activities and sharing the gospel with others.



Dr. Tom Miller (tkm@ncsu.edu) is professor of Electrical and Computer Engineering and associate dean for Distance Education and Information Technologies in the NC State University College of Engineering. He has been involved with Linux since around 1993, and helped organize the first Linux Expo at NC State in 1994. He developed what is believed to be the world's first X Windows spreadsheet in 1988. (Anyone remember X Windows version 10?) In 1990, he co-founded X Engineering Software Systems to develop the Xess spreadsheet and, more recently, GreyTrout Software's NExS spreadsheet.



Charles Price (ceprice@uncc.edu) started his career as a research physicist and is currently an assistant dean in the College of Engineering at the University of North Carolina at Charlotte where he manages the college's academic computing network. He builds wood furniture in his spare time.

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

[Advanced search](#)

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

Automated Installation of Large-Scale Linux Networks

Ali Raza Butt

Jahangir Hasan

Issue #78, October 2000

Need to load Linux on 100 workstations? Learn some tricks and techniques that could save you days of tedious work.

Installing Linux on a PC has been long considered a programming guru's domain. It usually takes a novice user weeks or even longer to get the system properly configured. However, with emerging installation techniques and package management, especially from Red Hat, Linux is on the verge of becoming user-friendly. Yet, even with these newer methods, one aspect of Linux that is still frustrating is installation on a large scale. This is not because it is difficult, but rather because it is a monotonous and cumbersome endeavor. The object of this article is to discuss the basics of a technique that will simplify large-scale installation. Furthermore, a scheme is also discussed for the automatic switch-on of a LAN employing Wake-on-LAN technology.

Installation Automation: Why?

A standard Linux installation asks many questions about what to install, what hardware to configure, how to configure the network interface, etc. Answering these questions once is informative and maybe even fun. But imagine a system engineer who has to set up a new Linux network with a large number of machines. Now, the same issues need to be addressed and the same questions answered repeatedly. This makes the task very inefficient, not to mention a source of irritation and boredom. Hence, a need arises to automate this parameter and option selection.

The thought of simply copying the hard disks naturally crosses one's mind. This can be done quickly, and all the necessary functions and software will be copied without option selection. However, the fact is that simple copying of hard disks causes the individual computers to become too similar. This, in turn,

creates an altogether new mission of having to reconfigure the individual settings on each PC. For example, IP addresses for each machine will have to be reset. If this is not done properly, strange and inexplicable behavior results.

What About Kickstart?

Those of us who have worked with Red Hat Linux are probably aware of the fact that it already offers a method of automated installation called Kickstart. This useful feature actually forms the foundation of the methodology we have developed. Kickstart allows us to specify beforehand our answers to all the necessary questions asked during the installation process. The specifications of a desired installation are listed in a special file called `ks.cfg`. This file is then placed on an NFS server and the target system is booted using a floppy disk. The setup prompt on the Red Hat distribution allows you to choose from a number of installation methods. Kickstart can be chosen as the desired technique by simply entering "ks" at the prompt. If everything has been done properly, *voilà!* The only message you will get at the end is the declaration of a successful installation.

Our Scenario

We were given the task to set up a Linux laboratory of sixty-four Pentium III machines connected via a 100MB Ethernet. Sixty machines were to be set up as workstations and four as various servers. With such a large number of machines, it was clear that a powerful means of installation was sorely needed. The power of our technique is evident from the fact that the whole setup process took us about sixty hours, spread out over fifteen days. Let's take a detailed look at the method we adopted.

Giving Kickstart a Kick!

The sixty computers obtained for use as workstations in our laboratory (see Figure 1) had hard disks but no floppy drives. To get Kickstart running, we needed to remove the case and manually connect a floppy to each machine, boot the machine, install Linux and, finally, remove the floppy. This is a long procedure since floppies go bad all the time and, even if they do not fail, it takes a minute or two waiting for the floppy to load. This can turn into an unpleasant two minutes as you wait with your fingers crossed, watching the screen, just to get the dreaded "Boot failed" message. Moreover, if a disk does go bad, it takes even longer to write another image onto a new disk.



Figure 1. The Linux Laboratory

A wiser approach had to be adopted. We merged the Red Hat Installation disk with a very fine net-booting package, etherboot, to obtain a network-bootable image of the disk. Now, since we also placed this image on a NFS server, only a 16KB loader was needed on the floppy which would boot up in under twenty seconds. This loader would then retrieve the actual image over the network. A new floppy could easily be made in less than thirty seconds.

The loader is, in fact, a ROM image; hence, to make it even more reliable we burned it on an EPROM. The Red Hat boot-disk image for network installation was kept on a DHCP/TFTP server. To get the installation running, the ROM was plugged in to the network card and the machine booted from the network. The same ROM can be reused to boot other machines. As the ROM is robust and small, an efficient way was thus developed for getting the installations running. We call this super-Kickstart.

Preparing the Tools

Before we could embark on setting up super-Kickstart, we needed to obtain and set up some software packages. The first was the etherboot package (see Resources). To serve our purpose, the package had to be modified a little as follows.

In the directory etherboot/netboot/mknbi-linux, edit the file mknbi.h as shown in Listing 1.

Listing 1

Now, edit the configuration file for etherboot, etherboot/src/Config.32, as follows. Locate the line:


```
CFLAGS+= -DDHCP_SUPPORT -DMOTD -DIMAGE_MENU
```

and change it to:

```
CFLAGS+= -DDHCP_SUPPORT
```

If the target machine has a BIOS that does not configure the network card properly, you may also need to append this line with **-DCONFIG_PCI_DIRECT** before compiling the packages.

Next, we moved to the top etherboot directory and did a **make all** to compile all the binaries.

Then, we created a directory to hold all the necessary executables for this setup. We copied the file etherboot/src/floppyload.bin and the appropriate ROM images, .rom and .lzrom, from the etherboot/src-32/ directory to this location. The file mknbi was also copied from the etherboot/netboot/mknbi-linux/ to this directory.

The second required tool was the streplace utility (see Resources). This useful package was utilized for replacing strings in files while configuring host-specific parameters that change with each workstation, e.g., host name and IP address. After compiling the binary, it was also copied to the working directory mentioned above. With these tools in hand, we happily moved on to the next step.

No More Installation Disk

A closer look at the Red Hat installation disk reveals that it contains a Linux kernel, an initial ramdisk image and some configuration files. For our purposes, we utilized only the kernel and the initial ramdisk image. To have a look at the contents of the disk image, mount it as a loop device using these commands:

```
mount -o loop  
cd mount_point
```

We then copied the kernel image (vmlinuz) and the initial ramdisk (initrd.img) to the directory we created earlier. In addition, the file syslinux.cfg provided the kernel options necessary for initiating a Kickstart install. They were noted. We had no further use for the installation disk beyond this point.

Setting up a Kickstart Option File

The Kickstart HOWTO discusses the syntax of the ks.cfg file in detail. Although very informative, it takes too long to generate this file. Therefore, the method we devised was to first install Red Hat Linux 6.1 on a machine using the "normal" CD-ROM method. All packages, options and settings for our to-be-

target-machine were manually specified. Once the system was up and running, it was tested for optimum performance and then used as prototype for the rest of the installations.

A special package called mkkickstart also had to be installed. The mkkickstart utility can extract information from an installation and print it on the standard output. We used it to do exactly that:

```
mkkickstart >ks.cfg
```

Any Kickstart installation that is now run with ks.cfg as the configuration file will create a replica of our prototype workstation. We did some minor editing of this file to implement some changes. Listing 2 is a sample from the start of the file.

Listing 2

Post-Install and Customization

The Kickstart technique offers provisions for executing any necessary post-install procedures needed once the installation is complete. This feature, besides allowing individual customization, is particularly useful when packages other than those included with the standard Red Hat distribution are to be installed. In our case, these included JDK (Java Development Kit) for Linux, among many others. We added the following lines to the post-install section and created a separate script and Perl program (see Listings 3 and 4) that would execute when the Red Hat installation had finished:

```
%post
cd /root
tar -xvzf install:/kickstart/install.tar.gz
cd installfiles
./postinstall
cd /root
rm -rf installfiles
```

The tar file (install.tar.gz) was placed on the Installation Server (install), from where it could be retrieved and executed to customize the system. Our special customization included un-tarring JDK from our ftp server, setting up linuxconf for web access, specifying the DNS server and allowing root remote shell access of workstation from the servers.

Listing 3

Listing 4

Setting up the Network Boot

One question that remained was where and how to place the ks.cfg file so that the target system was able to receive it even after it had undergone a DHCP/TFTP boot. An analysis of the installation procedure revealed that the tmp directory within the initial ramdisk is one of the locations that the Kickstart system looks for a configuration file.

The procedure of copying each ks.cfg to the appropriate location and then adding a kernel to initrd to make an encapsulated chunk of code was all performed by a script called superkick. A look at this script (see Listing 5) will show the steps involved in setting up the network bootable image.

Listing 5

We wrote another script, doitfor, to automatically customize the ks.cfg file and a post-install file for every workstation. It is shown in Listing 6. The major task that this script performed was inserting a specific host name and IP address within each ks.cfg using the streplace utility. This script takes as input the host name and IP address and generates a boot image to be uploaded using DHCP/TFTP boot.

Listing 6

DHCPD Configuration

To get the installation running, we needed to set up either a DHCPD server or a BOOTP server. The problem with the BOOTP method is that a list of network card MAC addresses has to be provided to issue IP addresses to the target machines. With a large number of installations, this would be quite a tedious job, so we opted for DHCPD. There is a long list of DHCPD options but we needed only the most basic ones, namely the default name-server address, starting IP and domain names.

Now the tricky issue here was that if we booted a machine an IP address would be issued to it. As long as it stayed on, the IP address would not be reused and any subsequent machine that was switched on would be automatically granted the next IP address by the DHCP daemon. If the first machine was turned off, then its IP address became free. Since DHCP has the liberty to assign any unused IP addresses, there is a good chance that the same IP address would be given to another new machine. If an installation were then run, this new machine would end up with the same host name and IP address as the one that was turned off. Simply put, they would have the same network settings and would not work properly.

To solve this problem, we changed the first IP address that the DHCPD was allowed to offer. This was done by reconfiguring the `/etc/dhcpd.conf` file (see Listing 7) and restarting the DHCPD. The `doitfor` script includes the code to carry out this task every time a boot image for a new target is created. Once an IP address had been allotted, DHCPD could be switched to the next IP address and the next installation started without complications.

Listing 7

The Installation Process

Once the DHCPD had been configured to offer the desired IP address for a target, we could proceed in two ways. The first was to burn the ROM image into an EPROM and plug the EPROM in to the network card. This obviously requires an EPROM programmer. The second and simpler way was to use the command

```
cat floppyload.bin <yourcard.rom> >>/dev/fd0
```

to create a bootable floppy disk carrying the ROM image and thus get the installation running. Some initial installations were carried out with the floppy method. Later, the availability of an EPROM writer allowed us to employ the EPROM technique, which worked fine and was good for experience and experimenting.

Note that completing more than two or three installations at a time overloads the network and brings down the efficiency. With two machines installing concurrently, we were able to achieve complete installations in an average time of fifteen minutes. The exact details of our installation procedure now follow.

First, we ran the shell script, `doitfor`, on the server to specify which machine we would be installing next. The floppy drive or the EPROM was plugged in to the target system and the system booted. The boot image was automatically retrieved, and the installation process started. While this process continued, we moved on to the next machine by running the `doitfor` script with new arguments corresponding to the next target. Booting the next machine would result in two installations running simultaneously. As soon as the installation on the first machine completed, we could start at the third one, and so on. The only hassle was plugging in the ROM/floppy and booting. Nevertheless, it was much faster than the standard method of using the floppy and manual settings.

If, on booting, the MAC address of the detected network card is reported as **FF:FF:FF:FF:FF:FF**, it is an indication that the network card is not initialized properly. There are two ways to overcome this. One is to switch off the Plug and Play OS feature of the motherboard during initial setup, forcing the BIOS to configure the network card. Note that switching this feature off is required only

during the installation bootup; it can be switched on later, if required. The second method is to enable the **-DCONFIG_PCI_DIRECT** option in the configuration file of the etherboot as discussed earlier.

Time Synchronization

With files shared among a large number of workstations, it becomes imperative that machines have their clocks synchronized so that file time stamps are globally comparable. Time synchronization helps in maintaining logs, updating and distributing updates, etc. We simply set up all the machines to match their time to that of a reference server at every startup by utilizing the `rdate` utility. To enable this on the client side, we simply added the following line to the `rc.local` file:

```
rdate -s
```

On the server side, the default time service had to be enabled in the `/etc/inetd.conf`. This was done by locating and removing the `#` sign from the beginning of the following line and restarting `inetd`:

```
#time  stream  tcp    nowait  root    internal
```

Fancy time-servers such as `NTPD` or `TIMED` could also have been used, but `rdate` works well for an undergraduate laboratory without causing headaches due to cryptic configuration issues.

Root Password Uniformity

Many volunteers helped with the installation. Once the installations were complete, security reasons demanded that we change the root password on every machine. This meant that either Linuxconf Web access was used for all the machines, or the system administrators had to manually change the password on each workstation. We wrote a script, `change_password_on_all.pl` (see Resources), to achieve this password change from a single server. The script requires that all workstations be configured to allow remote shell root access from the server using the `rsh` utility. This provision was configured during the post-install steps. It should be noted, however, that although this is not a very secure method of exchanging crucial information, it worked fine in our controlled laboratory environment. In addition, if caution is exercised by running the script only when the LAN is not being used, this method can prove to be very useful.

Startup/Switch-off Automation

Now we move on to a discussion of how we automated the LAN startup and shutdown by controlling it from a single machine. To understand this we have

to take a closer look at AMD's Magic Packet technology. Many network cards now come equipped with the Wake-on-LAN feature. This means that when the machine is switched off (i.e., the ATX casing has power but is not in the "on" state), it provides a small amount of power to the Wake-on-LAN enabled network card through a three-wire header connecting the motherboard and the card. Hence, the card is actually alive and able to keep watch on the LAN for a special packet, called the Magic Packet (see Resources). On receiving such a packet, the network card generates a pulse that can be used to switch on the machine. The Magic Packet is, in fact, a stream of 0xFF characters followed by the MAC address of the NIC, repeated a specific number of times. The probability of such a packet occurring coincidentally during normal operation is very rare. Hence, the reception of such a packet can be safely assumed to indicate that it is indeed meant for the target machine.

We wrote a Perl program named **switch** (see Listing 8) that switches one or all machines on the LAN either on or off. It generates a Magic Packet and broadcasts it over the network. The target machine, on receiving this packet, switches on. To switch a machine off, a straightforward remote shell invocation of **halt** or **shutdown -h** suffices. To make things more manageable, we made the script parse `/etc/switch.conf` for information regarding which host on the LAN has what MAC address. It should be obvious that the MAC addresses are required only for switching on and not off, as the shutdown invoked through the remote shell requires only target IP addresses.

Listing 8

A small drawback is that, although the switch-on is possible whether the target machine runs Linux or not, the switch off is possible only if the target machine properly boots into Linux. This method does not provide for switching the system off if the target machine fails to boot properly. It should, however, be noted that if a machine does fail to boot properly, it would require manual attention for fixing it anyway.

In our laboratory, we had Intel 440BX2 motherboards with D-link network cards that support the Wake-on-LAN feature. The motherboard requires that the feature be switched on from the BIOS. There is a large range of other component brands that support this feature as well. Chances are that if you have purchased your hardware in the last year, you already have these features.

Server Configuration

We utilized standard procedures in setting up the server configuration. We set up the DNS, NIS, FTP, Apache, time and NFS services. One special consideration was that no two services were provided by a single IP address. Although we had

only four actual servers, we relied heavily on IP aliasing to create virtual personalities for each service. This aliasing method provides for transparent shifting of the services from one machine to another in case of a failure, providing some degree of fault tolerance. This approach is in continuation of our previous work on network fault tolerance, reported in *Linux Journal*, June 2000.

autofs

autofs is a kernel-assisted auto-mounter for Linux which allows the system to dynamically mount a file system on demand. It is like using MS Windows, where, when you need to access a floppy drive, you do not have to specifically attach a drive to a mount point. For example, if autofs is configured to auto-mount a CD-ROM at, say, mount point `/misc/cd`, then every time a CD-ROM is inserted into the drive and the directory is changed to `/misc/cd`, the CD-ROM will be automatically mounted at this point. If the mount point is not used for a while, it will be automatically unmounted.

autofs was found to be very useful for our scenario. Over the span of one week, we have many different classes coming into the laboratory. Keeping all the user files mounted on each workstation all the time created a lot of server load and network traffic—an inefficient and undesirable situation. We divided the file system into four groups and mounted them individually via autofs rather than hard-binding the NFS servers in the `/etc/fstab` file. This reduced the server load to a quarter of its original.

To maintain flexibility, we used NIS for the autofs maps. The map `auto.master` provides information about the mount point of the autofs system, and `auto.home` gives information about what file system should be mounted and from which server. It was discovered that the autofs does not check for an NIS map if the file `/etc/auto.master` is present. Hence, to make it work properly, we removed the file `/etc/auto.home` from all the workstations that were going to employ autofs. To include these maps in the NIS database, select the rule `auto.home` and `auto.master` in the NIS Makefile located at `/var/yp/`.

The following line was added to `/etc/auto.master` on the server:

```
/home auto.home --timeout 60
```

and these lines were added to `/etc/auto.home` on the server:

```
#mount point options source host+path
g1 -rw,hard,intr nfs1:/home/g1
g2 -rw,hard,intr nfs2:/home/g2
```

to enable building the proper database.

Resources and Acknowledgements



Ali Raza Butt, on the left, (ali@ieee.org) has recently graduated from the Department of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan. He joined the doctoral program in Electrical and Computer Engineering at Purdue University in fall 2000.

Jahangir Hasan, on the right, has recently graduated from the Department of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan. He joined the doctoral program in Electrical and Computer Engineering at Purdue University in fall 2000.

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

[Advanced search](#)

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

[Advanced search](#)

Graphics: Pick a Card...Any Card

Matt Matthews

Issue #78, October 2000

With graphics capabilities being so important and new cards appearing all the time, you need a score card to pick the right one. Here it is...

The winter of 1999 was a turning point for 3-D games on Linux. Two of the highest profile first-person shooters, Epic's Unreal Tournament and id Software's Quake III Arena, were released nearly simultaneously for Windows and Linux. As little as a year earlier, the state of 3-D graphics acceleration on Linux would have prevented the platform from reaching such parity with the Windows world. Now, nearly all major video card makers are working to ensure that their cards provide some level of 3-D graphics support on Linux. With that hardware support, Linux gamers are presented with a choice hitherto unheard of: picking the perfect 3-D video card from over a dozen choices.

The Players

In the beginning, Linux users had only one choice in hardware acceleration: 3dfx. Through the work of Daryll Strauss, the line of Voodoo cards have been well supported for quite a while, providing both Glide and OpenGL (via Mesa) acceleration. Currently, 3dfx has contracted Precision Insight to complete XFree86 4.0 drivers that use their Direct Rendering Infrastructure (DRI) for 3-D hardware acceleration.

With the release of hardware specifications for their G200 and G400 cards, Matrox has recently shown strong support for the Linux sector. Not only are their cards well supported for 2-D acceleration, but the work of the Utah-GLX team has provided robust 3-D acceleration of OpenGL (again via Mesa) in XFree86 3.3.x. For the new XFree86 4.0, Precision Insight has been contracted to maintain 3-D acceleration for the Matrox G400 via the DRI.

ATI, the biggest supplier of graphics cards worldwide, has also shown support for Linux recently. Besides assisting the Utah-GLX project to support the 3-D

portion of the older Rage Pro cards, they are also relying on Precision Insight to provide acceleration for their Rage 128/Rage 128 Pro cards in XFree86 4.0 via the DRI. Further, ATI has promised Linux drivers for boards based on their brand new Radeon chip.

Finally, NVIDIA has made a big splash in recent months with their release of XFree86 4.0 drivers supporting a wide array of cards, from the TNT all the way up to the newest GeForce 2. Their OpenGL support and speed, all developed in-house, is unrivaled in the Linux world and is currently setting the standard for speed and OpenGL conformance.

Together, these companies provide a selection of about a dozen cards that support 3-D acceleration in Linux. We'll take a quick look at each card and point out its strengths and weaknesses.

The Cards, Company by Company

Now, let us take a look at some of the most popular 3-D cards supported under Linux. We've grouped them by manufacturer so that we can make general comments about the company and their hardware.

3dfx

The 3dfx line of cards starts with the low-end Voodoo1 and stretches up to the very newest Voodoo4 and Voodoo5 cards. Along the way, you can see the evolution from the limited add-on 3-D-only cards like the Voodoo Graphics and Voodoo2 cards to the fully-featured 2-D/3-D of the Voodoo5.

All 3dfx cards natively support a 3-D graphics API, developed by 3dfx, called Glide. Glide is used in a handful of Linux games like Loki's Myth II and Epic's Unreal Tournament. For such games, 3dfx cards are often the best (and sometimes the only) choice for hardware acceleration. Unfortunately, most 3dfx cards also lack crucial features that some gamers want, like full support for 32-bit color and a stencil buffer. The newest Voodoo4 and Voodoo5 cards have finally added these features, as well as others.

Finally, for OpenGL support, the 3dfx drivers depend on Mesa, an open-source implementation of the OpenGL API. This means that the OpenGL performance and conformance of the 3dfx cards is dependent to a great degree upon the development of Mesa.

Voodoo Graphics (a.k.a. Voodoo1)

Price: Around \$30 or less used (not available in retail any longer)

Benefits: The relatively low price and native support for Glide are the primary advantages of the Voodoo Graphics board.

Disadvantages: The Voodoo Graphics board is not an all-in-one graphics board; it provides only 3-D acceleration while letting your main 2-D board take care of normal operation. Further, it provides acceleration up to a resolution of only 640x480 due to limited RAM.

Voodoo2 and Voodoo2 SLI

Price: Around \$80 for a new Voodoo2-1000 from 3dfx

Benefits: Still a reasonable budget accelerator, the Voodoo2's real power comes when working in pairs to provide Scan Line Interleave (SLI) functionality. In this configuration, each card renders half of the scan lines on the screen, increasing performance and the maximum resolution possible in 3-D from 800x600 for a single card to 1024x768. Glide support is also a bonus.

Disadvantages: Like the Voodoo Graphics, the Voodoo2 is an add-on card to supplement your 2-D card. That means that for SLI you could be using as many as 3 PCI card slots just for video cards. Coupled with the relatively high price and aging technology, even the Voodoo2 SLI setup looks less than appealing.

Banshee

Price: Around \$70

Benefits: The Banshee is a 2-D and 3-D card in one, making it a step better than the Voodoo2 and Voodoo Graphics cards. The low price and the native Glide support make the Banshee a reasonable, cheap, low-end card.

Disadvantages: The Banshee, while clocked higher than the Voodoo2, has only one texture processor. So it is faster than a single Voodoo2 in some games, but can be slower in those that require multi-texturing (which means just about every newer game).



Figure 1. Voodoo3-3000

Voodoo3-2000, 3000, 3500TV

Price: From \$90 for the Voodoo3-2000 to \$200 for the Voodoo3-3500TV

Benefits: The Voodoo3 provides more 3-D power than the Voodoo2 SLI along with a 2-D core similar to the Banshee for a reasonable price. Additionally, with several models and PCI options, there is a card to fit almost every need and budget. Further, the 3500TV has a TV tuner that is supported under Linux.

Disadvantages: For the best performance, you'll need to get a 3500TV and that is a bit pricey. Further, the lack of modern features that the Voodoo4/5 and competition offer is disappointing.



Figure 2. Voodoo5-5500

Voodoo4 and Voodoo5

Price: From \$179 for the low-end Voodoo4 to \$600 for the Voodoo5-6000

Benefits: All the advantages of the Voodoo3 with more 3-D performance and features like 32-bit color, large texture support, and stencil buffers. Based on a

scalable architecture around the VSA-100 chip, 3dfx is again offering a range of cards, in both PCI and AGP, to appeal to all customers. 3dfx is also pushing the ability to do full-scene anti-aliasing (FSAA) with hardware support in the V4/5 cards. This would improve the visual quality of all games (OpenGL and Glide) without the performance hit normally associated with FSAA.

Disadvantages: The price for FSAA is tremendous amounts of RAM, like the 64MB on the Voodoo5-5500, and that may cost a pretty penny. The best V5 card, the Voodoo5-6000, will probably be out of the reach of most gamers.

Matrox

Normally known for their high-quality 2-D core, the latest Matrox cards, like the G400Max, are also powerful 3-D performers. Now that they are releasing the hardware specification and working with the Utah-GLX team and Precision Insight, those 3-D benefits can be seen in Linux as well. Like the 3dfx and ATI, Matrox cards depend on the Mesa implementation of OpenGL.

G200

Price: \$50 or less

Benefits: Relatively speaking, the G200 is a cheap card that can provide reasonable hardware acceleration and sharp 2-D. Despite being a lower-end card, it does support 32-bit color in 3-D if you absolutely want it.

Disadvantages: Like its price reflects, the G200 is a low-end card. Don't expect to play Quake III Arena in all its glory at anything close to an acceptable frame rate.

G400

Price: \$100 for a G400 single-head to \$200 for a G400Max dual-head

Benefits: The G400Max is quite fast with the Utah-GLX drivers, and in a dual-head configuration can drive two monitors at once. Fortunately, the G400 comes in several models to suit different needs and budgets. Further, the G400 supports modern 3-D features like 32-bit output and large textures and can maintain a solid frame rate while doing so. Add in the clean 2-D, and the G400 cards look appealing.

Disadvantages: To get the best performance, expect to lay out some cash for the G400Max. It would also be nice if Matrox helped support bump-mapping in Linux like it does in Windows.

ATI

Like Matrox and 3dfx, ATI is helping the Linux community by working with developers to create and maintain their drivers. Further, they are looking into more than just 3-D by trying to add 2-D video acceleration (as might be used with DVD players) to their Linux support. ATI also has their Radeon chip on the horizon, with its plethora of new features and promised speed and Linux drivers. Their support with the Rage Pro and Rage 128 could be just the precursor to the splash that ATI could make in the Linux world.



Figure 3. ATI-RagePro

Rage Pro

Price: \$35

Benefits: Now a very old card, the Rage Pro can be had very inexpensively. The support with the Utah-GLX drivers is as fast, if not faster, than the drivers for Windows.

Disadvantages: The Rage Pro just can't keep up with the rest of the pack. Quake III Arena is playable only after all the special effects are turned off. This should be considered the very low-end for any hardware acceleration in Linux.

Rage 128, Rage 128 Pro

Price: \$60 for the Rage 128 to \$130 for the Rage 128 Pro

Benefits: The Rage 128 line of cards are now supported under XFree86 4.0 and take advantage of the DRI for hardware acceleration and should perform quite well for the price.

Disadvantages: The drivers are still in development and aren't as fast or as optimized as those offered by other cards.

NVIDIA

Somewhat of a late bloomer relative to the others, NVIDIA finally made their presence felt in Spring 2000 when they released their XFree86 4.0 drivers. Based on the same code as their Windows implementation, the drivers offer the fastest and most conformant OpenGL acceleration available in Linux. Further, NVIDIA supports a wide range of cards, from their venerable TNT all the way up to their latest GeForce 2 cards. This range of cards supports many features, like 32-bit color and stencil buffers.

One special feature available on the GeForce and GeForce 2 cards is the Graphics Processing Unit (GPU). This GPU offloads some of the computationally expensive work (mainly coordinate transformations and lighting operations) normally done by the CPU, leading in some cases to improved performance on lower-end systems. Additionally, NVIDIA is pushing support for all features currently available in Windows, including AGP transfers, texture compression, and flat-panel output as well as others. They also promise to offer in Linux the same full-scene anti-aliasing that has recently arrived as part of their Windows drivers.

One other advantage that NVIDIA has over 3dfx, Matrox and ATI is their business model. Currently they just make chips and other companies make the boards. The competition among the board makers can then lead to better prices for consumers.

TNT

Price: \$30 used (not available in retail)

Benefits: A cheap, older card, the TNT offers solid OpenGL acceleration and 32-bit color.

Disadvantages: As the price might suggest, this card looks slow in today's market and doesn't have as much memory as newer cards. It is becoming a low-end card.

TNT2 and TNT2 Ultra

Price: \$40 for a smaller TNT2 up to \$140 for a TNT2 Ultra

Benefits: A range of cards to choose from, with varying speeds, a gamer on a budget can find a TNT2 to suit his needs. The TNT2 Ultra is still a reasonably

speedy card and with its full set of features, solid OpenGL support, and 32MB of RAM, will last a bit longer for playing the newest games at a passable frame rate.

Disadvantages: The price for the high-end TNT2 Ultra can seem a big pricey for its power with today's games. And some features can be used only at a sizeable performance hit (like 32-bit color).

GeForce SDR and DDR

Price: \$130 for the GeForce 32MB SDR up to \$290 for the GeForce 32MB DDR

Benefits: The NVIDIA GPU can help boost the performance on a system with an older CPU. Further, the support for texture compression, 32-bit color, and many other features makes the card a versatile OpenGL accelerator.

Disadvantages: Unfortunately, even with DDR RAM the GeForce is bandwidth limited on high-end systems. Currently, the benefits seen from using the GPU are limited to just a few games. And expect to pay well over \$200 for the best card.

GeForce 2 GTS

Price: Over \$300

Benefits: The fastest card currently available on the market, and with the GPU it should last even the demanding gamer for a good long time. This is the card to have for today's hard-core Linux gamer. And with several companies making boards, there should be competition to bring prices down. This is also the NVIDIA card with enough bandwidth to support FSAA when NVIDIA adds that feature to their Linux drivers. This is the most future-proof card you can buy.

Disadvantages: For the fastest card on the market, expect to pay dearly; the GeForce 2 can cost as much as the CPU in your system. Also, there still seem to be some bandwidth issues, especially on higher-end systems at high resolutions. Finally, support for the features of the GPU haven't yet been fully realized in today's games.

How to Choose?

Now that you can see all the cards available, here are few questions that may help you choose the card that best fits your needs.

- Is fast, accurate OpenGL performance important to you? Do you demand the best Quake III Arena performance?

If you require the best OpenGL acceleration under Linux, look to NVIDIA for your card. They have a great OpenGL implementation and can offer you fast hardware as well. That carries over to Quake III Arena, where OpenGL performance is critical.

- Do you play Unreal Tournament or other games that require Glide?

Currently, Unreal Tournament plays best in Linux using Glide. That means you need 3dfx hardware, and for best performance, a Voodoo3 or better. While it is possible that the OpenGL renderer in UT will catch up to Glide, it's by no means a certainty.

- Do you want to run two monitors at once without sacrificing a card slot for the extra video card?

The Matrox G400 dual-head cards offer reasonable OpenGL acceleration through Mesa and the ability to drive two monitors. Other companies allow you to run several cards at once in your system, but Matrox's solution is more elegant.

- Is your budget limited and you need a 2-D/3-D card with reasonable performance?

The ATI Rage 128 cards are a good balance of features and price, while the NVIDIA TNT2 comes in a close second.

The best way to find out what you like, naturally, is to see the products in person. So while you can't buy each card to try it out, a next best choice is to ask around your local LUG and find a user with a card you are considering. Ask what they like about it and perhaps even ask to see their system up close; in doing so, you'll not only get an idea of what value you see in the hardware, but also what an owner has come to think about it as well.



Matt Matthews, a PhD student in Applied Mathematics at NCSU, became a Linux user in the summer of 1999 when he and his advisor bought new machines for their research. In addition to his computational work, he enjoys playing games and testing out the newest 3-D video cards under Linux. When time allows, he writes up those experiences for his favorite web-site, LinuxGames (www.linuxgames.com) can find him out with his wife, Mandy, or tending to his ever-growing collection of video games.

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

[Advanced search](#)

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

[Advanced search](#)

BusyBox: A Swiss Army Knife for Linux

Nicholas Wells

Issue #78, October 2000

Learn how to save disk space with this open-source tool for embedded systems.

Linux is being used in more and more tight places—devices or situations where multiple gigabytes of storage or dozens of megabytes of RAM are simply not available. Many *Linux Journal* articles have addressed this growing embedded systems space. Here, I will describe a terrific open-source tool for people needing a suite of utilities for use in these tight places.

BusyBox is a single-binary application that includes the capabilities of more than 70 standard Linux utilities. The BusyBox project was started by Bruce Perens when he was leading the Debian project. Many others have contributed code since then; the project is currently maintained by Erik Andersen, who also maintains a web page for it at <http://busybox.lineo.com/>. (BusyBox is sponsored by embedded Linux vendor Lineo, Inc. See <http://www.lineo.com/>.)

The corresponding standard utilities that BusyBox replaces occupy about 1.5MB of disk space in a standard Linux system. The BusyBox program uses only 260KB of disk space; it can even be compiled to include only a handful of utilities that you select, still within a single binary. We'll see how to do this later.

How It Works

Linux distributions normally include packages with many separate utilities, often in sets named `textutils`, `shellutils` or something similar. These utilities are generally very full-featured. For example, the command to list files, `ls`, supports over 50 command-line options. Because these utilities are so full-featured, they are sometimes larger than you might expect. For example, the `ls` command (dynamically linked) occupies 48KB of disk space. While you wouldn't notice that on a standard PC, many of those 50 options are just wasted space on an embedded system or boot disk. BusyBox combines numerous utilities in a

single binary, limiting the features of each utility to those most commonly needed. The **ls** command in BusyBox has a mere 12 options—more than enough for most of us.

When you enter the name of a utility in a shell (at the Linux command line), the shell locates the binary with that name and executes it. When you install BusyBox, it creates links in your file system so that instead of having a real **ls** command, you have a symbolic link named “ls” that refers to the BusyBox binary. A directory listing of **ls** would then appear like this:

```
lrwxrwxrwx 1 nwellls users 12 May 17 14:47 ls ->/bin/busybox
```

Typing **ls** at the command prompt causes the shell to launch the BusyBox program, which in turn examines the name under which you invoked it. Seeing that you entered **ls** in this example, the BusyBox program proceeds to act like **ls**, interpreting any additional options according to that command.

What Does BusyBox Include?

If this talk about 70 utilities in one is making you curious, here are a few samples of what BusyBox includes. The web site given above has a complete list with documentation:

- **chmod, chown, chroot, lsmod, rmmmod, insmod**
- **dd, df, du, mkfs.minix, fsck.minix, mount, sfdisk**
- **find, grep, sed, gzip, gunzip, tar**
- **kill, killall, ps**
- **cp, cut, mv, rm, ln, more, tr**
- **nslookup, ping, telnet**
- **init, syslogd, sh**

Many of these utilities are reduced versions to fit in the 260KB total space mentioned above. For example, the **sh** shell doesn't support if/then/else statements or while loops, but it has enough built-in commands to handle simple startup shell scripts. Also, commands like **init** and **tar** have greatly reduced functionality, but they are sufficient to get the job done for many applications where you don't have space for the full-blown utility.

Compiling BusyBox

Adding BusyBox to your system is simple. First, obtain the source code by visiting <ftp.lineo.com/pub/busybox> and downloading the most recent version of the compressed tar file. As of this writing, the most recent version is the file

busybox-0.45.tar.gz, but newer versions will probably be available by the time you read this. Work on BusyBox seems to progress at a steady pace.

After you've downloaded the tar file, place it in a working directory and use this command to untar the file:

```
tar xvzf busybox-0.45.tar.gz
```

Change to the newly created busybox directory (I write the name in lowercase this time to match the binary name) and enter the **make** command:

```
cd busybox-0.45
make
```

Now you're ready to test a few BusyBox commands. The busybox binary is located in the main busybox directory after you use the make command. In order to use a particular feature of BusyBox, execute the busybox binary with a command name as a parameter. For example, to use the **ls** command, from within the busybox-0.45 directory, enter:

```
./busybox ls
```

Or to use the **lsmod** command, enter:

```
./busybox lsmod
```

Additional command options can be placed after the command name. For example, to use the **du** command to view only the contents of the /etc directory tree, type this: **./busybox du /etc**

Another important feature of BusyBox is that you can see a small on-line help screen for each utility that BusyBox replaces. Just use the **--help** option with the command. For example, to learn about the ls options that BusyBox supports, enter **./busybox ls --help**

This displays the following help text:

```
BusyBox v0.45 (2000.05.17-20:38+0000) multi-call
binary -- GPL2
Usage: ls [-1acdeInpuxACF] [filenames...]
Options:
-a      do not hide entries starting with .
-c      with -l: show ctime (the time of last
        modification of file status information)
-d      list directory entries instead of contents
-e      list both full date and full time
-l      use a long listing format
-n      list numeric UIDs and GIDs instead of names
-p      append indicator (one of /=@|) to entries
-u      with -l: show access time (the time of last
        access of the file)
-x      list entries by lines instead of by columns
-A      do not list implied . and ..
-C      list entries by columns
-F      append indicator (one of */=@|) to entries
```

Installing BusyBox

BusyBox is most useful on a system without a set of regular Linux utility programs, but you'll probably be exploring it on a standard Linux system. Because of this, you should use the PREFIX variable when installing BusyBox. The installation process creates symbolic links for all the utilities that BusyBox supports. This allows you to enter **ls** instead of **busybox ls**. Suppose you had begun working with BusyBox in the **/tmp** directory (such that a directory called **/tmp/busybox-0.45** was created by the **tar** command). Then, if you want to create symbolic links in the same area, use this command:

```
make PREFIX=/tmp/busybox-0.45 install
```

The **/tmp/busybox-0.45** directory will then contain subdirectories named **bin**, **sbin** and **usr**, each with symbolic links to **/bin/busybox**. You'll also need to copy the busybox binary to **/bin** before using these symbolic links:

```
cp /tmp/busybox-0.45/busybox /bin
```

Now you're ready to explore the symbolic links in your BusyBox subdirectories. For example, change to the bin directory:

```
cd /tmp/busybox-0.45/bin
```

Then use the **ls** symbolic link with the **--help** option:

```
./ls --help
```

You see the same help text as indicated previously. This shows you that BusyBox is being used instead of the standard **ls** command on your Linux system.

BusyBox uses the GNU C library, or glibc, which can add substantially to the space requirements of an embedded system or boot disk. You might consider looking at alternate C libraries to save space. Examples include minix libc and newlibc. Another example that looks promising but doesn't yet support the full functionality of BusyBox is the uClibc project from Rt-Control (see <http://www.uclinux.org/>). The maintainer of BusyBox, Erik Andersen, is currently working to enhance this mini C library so that it can be used to reduce the total size requirements for BusyBox.

Configuring BusyBox

The description of BusyBox so far is straightforward, but doesn't capture all that the program offers. Returning to the source code you un-tarred when you compiled BusyBox, load the file **busybox.def.h** into a text editor:

```
cd /tmp/busybox-0.45 vi busybox.def.h
```

The first part of this file (about the first 100 lines) contains **#define** statements for each utility capability that will be included in BusyBox. If you don't want to include the capabilities of one of these utilities, simply comment out that line. For example, if you don't need **sed** in the system you are using BusyBox on, comment out the line for sed using two forward slashes, like this:

```
//#define BB_SED
```

Commenting out a few of the larger utilities greatly reduces the size of the final busybox binary. For example, removing five complex programs (**init**, **tar**, **sfdisk**, **gzip** and **gunzip**) reduces the size of the busybox binary from 260KB to 155KB.

The second part of the busybox.def.h file (after a few explanatory comments) contains **#define** statements that activate or disable various features of BusyBox. Some of these features are intended to save memory, such as eliminating the use of the /proc file system, reducing the amount of on-line help provided and eliminating the use of regular expressions. Other **#defines** are specific to features of a single command. For example, you can eliminate the ability to create new tar files with the tar features of BusyBox. Unless you really need to shave off a couple more KB in the size of BusyBox, you shouldn't need to alter the **#define** options in the second section of the busybox.def.h file.

Conclusion

Embedded versions of Linux such as Lineo's Embedix may be the most obvious use of BusyBox, but you might come up with many other uses. For example, if you need to create an initrd file to boot up a system with unusual hardware, you can use BusyBox functionality to add the basic system utilities with a single easy-to-manage binary. Or, you might use BusyBox as part of a boot diskette or single-disk version of Linux, as the Linux Router Project and the Debian boot floppies.

Resources



Nicholas Wells is a writer who has published about ten books on Linux, KDE, and the Web with Sybex, IDG, etc. He enjoys studying languages (the human kind) and traveling to exotic places for Linux conferences.

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

[Advanced search](#)

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

Contributing to the Linux Kernel—The Linux Configuration

Joseph Pranevich

Issue #78, October 2000

Learn the correct procedures for submitting a patch to the kernel.

The Linux kernel has always been one of the most prized gems in the Open Source community. Based around a philosophy of shared resources through modularity, it somehow is both well-written and written by committee. (Or, at least, by many individuals and teams which argue/agree over features.) One of the methods by which Linux keeps everything neat and modular is the kernel configuration system, often referred to as config, menuconfig and xconfig. These are the scripts that an installer of a source kernel must run in order to set up the kernel options, but you probably know that if you are reading this. On the outside, these look like three very separate programs with completely separate interfaces. In reality though, all three draw from the same fundamental rules that many programmers of the Linux kernel must know in order to spread their work or even to submit their patches to Linus. It is this fundamental system that gives Linux users the options they need to design a Linux system based on their needs.

Since the Linux kernel is an open-source project, it obviously accepts submissions from its users for new features. Often, however, programmers with the desire and the know-how to add features to the Linux kernel choose not to for a variety of reasons. In this article, I hope to clear up some of the mysteries surrounding the kernel configuration system that may be hindering users and keeping them from becoming developers. Every brain counts in open-source efforts, and every programmer who adds his or her changes into the kernel makes the kernel more robust for the rest of us.

What's in a Patch?

To start off with, there has to be a reason you are mucking about in the kernel configuration scripts in the first place. Maybe you are just exploring the system and awaiting the day when you too will be submitting patches to the kernel. Or

maybe (and more likely) you have added a particular feature to the kernel that you feel deserves some more widespread use, but you want to have it ready to integrate for Linus or Alan or another kernel-developing guru. For the purposes of this article, I will use a hypothetical patch to make the random device driver a compile-time option, although I should stress that in reality I had absolutely nothing to do with the creation of that driver. (This is the driver that controls the `/dev/random` and `/dev/urandom` devices.) Also, I will not be discussing in depth the creation of kernel modules in this text—I will assume you can extrapolate how to do it from this article, especially if you were smart enough to create a modularized driver in the first place.

Choosing Your Config Name

The first step in modularizing your program should be obvious: you need some name that the C preprocessor can recognize to help it sort out the differences between what changes are yours and what are not. The kernel handles this distinction through the use of preprocessor instructions: the **`#ifdef ... #else .. #endif`** constructs throughout the kernel.

The first thing to make sure of when you do this is to be consistent. In a system as complicated as the Linux kernel, a little bit of consistency can save a lot of headaches later. You should look in `Documentation/Configure.help` for similar options and check to see if they have a common prefix (after **`CONFIG_`** of course). For example, all block device options start with **`CONFIG_BLK_DEV_`**. This is relatively easy to change later, of course.

Once the name is selected, you should make **`#ifdef...#endif` blocks** around portions of the code that your patch changes (having a Linux tree around when you do this helps, as you can diff it and easily see what you changed). If you removed existing code, you'll need to integrate the `#else` blocks from the real tree, unless you were smart enough to keep them around while you were writing your patch. (I usually use the construct `"#if 0"` early in the programming stage.) At this point, compiling the kernel should work, and your option will be correctly disabled and we can continue to the next step. If it doesn't work or portions of your patch are still present, you obviously need to go back and double-check your diffs.

For the purposes of my example, I would choose the name `CONFIG_RANDOM` for the option. The random devices are character devices, and at the time of this writing, there was no **`CONFIG_CHAR_`** (or similar) prefix in common use.

The Configure Scripts (`Config.in`)

The next step we need to take is to add the new configuration option to the configure system. Fortunately, this is fairly easy with only a couple of warnings.

In the directory where you have the majority of your patch (in my example, drivers/char), there will be a file called 'Config.in' which contains the configuration options for the code in that directory. It is possible to put the config option in a different directory's config file, but it obscures the readability a little and may make it difficult to locate your code later. However, if it definitely belongs somewhere other than where you have it (or if the location of your code does not have one of those files), you should use your own best judgment and be prepared to move it later. Browsing through this file, you will see that it contains what appears to be a rough scripting language similar to Bash or another shell script. This scripting language is called, easily enough, "Config Language" and should not be terribly difficult to get your arms around. For the purposes of our non-modular example, we don't need to work with the full vocabulary of the language and can concentrate on only a few keywords (defined below). For a more complete guide to the language, a complete reference is provided with newer kernels (Documentation/kbuild/config-language.txt). There are plenty of examples provided in the actual configuration files, however, and the language is simple enough that a real understanding of the language and syntax is not required for normal maintenance. In general, lines in this file are formatted with one or more keywords (called verbs) followed by some arguments. Here is a partial list of verbs and their meanings:

- **comment**: An unparsed comment except when preceded by the **mainmenu_option** command which would cause the comment to be used as a heading.
- **mainmenu_option**: A verb that makes the next comment into a heading. I cannot explain enough how odd this seems to me.
- **bool**: Boolean (yes/no) configuration option. This verb is always followed by a question and a configuration variable to put the result ("y" or "n") in.
- **tristate**: A value similar to a bool but with the additional "make as module" possibility denoted as "m". This is applicable only to device drivers.
- **if... fi**: A conditional block that is evaluated only if a certain configuration variable is set.

The idea here is to locate the section of the file which corresponds to where you would like your option to be in the configuration programs. This should naturally be the location where your patch is most related to the other options provided. Once you have located where it would go, you should format a line similar to the following with your own information thrown in:

```
bool 'Random driver support' CONFIG_RANDOM
```

(Please note here that a ' is used as the quote character. This can be easy to miss.)

Config File for CONFIG_RANDOM

If your configuration option relies on another to be set, this model becomes more complicated. You will need to surround your options with an `if ... fi` block that tests the prerequisite option. There are a number of examples in the assorted configuration files to help you with this process; when in doubt, copy. One final word: you should be aware that this file is used not only to generate the selection lists in the configuration processes, but also to generate the `include/linux/autoconf.h` file. In order to preserve the readability of that file, you should be careful that options you add do not come after other subheadings or the configuration option will not appear in the right place when that file is generated. (Of course, it will still work no matter where it is in the file, but for readability, this is something to consider.)

Setting up the Default Configs

At this point, you have your patch surrounded by defines which correspond to configuration options. You can now toggle the use of your patch in the configurator. The hardest part is done, now we just need to fill in some loose ends.

As you have probably noticed if you have used Linux on more than one architecture, there are different options for each of the different platforms. Thus far, you have not concerned yourself with making much of that distinction, but now it is time. In order to modify the `defconfig` file the right way, you'll need to take a couple of steps. First, you should back up your `.config` file with your personal configuration settings. Replace that file with a copy of `arch/<whatever>/defconfig` and rerun the configurator. Select whatever option you would like to be the default for your config option, and save that file. Replace `defconfig` with the new `.config` and replace the old one. You now have a new default configuration file built for your architecture.

If your patch works for multiple architectures (for example, it's a generic network protocol type or something that is generally applicable), you'll want to make default config files for all the architectures it supports. This can be done most easily by repeating the steps above, except first changing the `ARCH` line in the source `Makefile` to reflect the architecture you are building the config file for.

The Help File

The final and often overlooked portion of a patch is the help file that must accompany the configuration option. The file in question is `Documentation/Configure.help`. The format of that file is something like this (or see the attached example):

```
<description>  
<variable name>  
<help file>
```

In addition, it should be noted that there must be an extra blank line between config files, and newer kernels support having blank lines in the help text itself, provided that the blank line actually contains a space or other white space.

It is highly advised that you attempt to locate a portion of this file that roughly corresponds to what your option is for. Often, but not always, options in similar places in the configuration menus will have similar places in the help file, and that can serve as a helpful indication as to the best place to put your particular piece. If you still do not know where to put everything, I advise sending e-mail to the current help maintainer with a question and a brief description of what your patch does (maybe even include the help text you would like to have added).

Conclusion

You should be done at this point. Before you can send off your patch for submission, it is wise to compile it both enabled and disabled and make sure both kernels work properly. If there are dependencies, make sure they are working correctly and that all the options you would expect to have, you have, and vice versa. And finally, you should send your patch to the kernel list and request bug reports before you announce it as being done to Linus and the gang, or be prepared for a heavy dose of reality, especially if you made changes that impact architectures you don't actually have.

[CML2: A New, Easier Kernel Configuration System](#) by Eric S. Raymond

email: jpranevich@lycos.com

Joseph Pranevich (jpranevich@lycos.com) is an avid Linux geek, and while not working for Lycos, enjoys writing (all kinds) and working with a number of open-source projects.

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

[Advanced search](#)

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

[Advanced search](#)

Configuring, Tuning and Debugging Apache

Reuven M. Lerner

Issue #78, October 2000

RPMs, mod_perl, apachectl, telnet, mod_status, DSO, apxs, Apache::Status...Scared yet? Mr. Lerner tells us about all of these so we can better manage our web servers.

As longtime users know, Linux, Apache and other free software packages are both more stable and more easily configured than their closed-source counterparts. However, this obviously does not mean that free software advocates are immune from software bugs and configuration problems. And indeed, the complexity of much open-source software can sometimes be frustrating; it's often unclear which option to modify, or even where to begin.

This month, we will look at some of the tools and techniques that webmasters can use when trying to configure, tune and debug their Apache configurations. This cannot be a comprehensive list, simply because so many things can go wrong. However, being able to identify the cause of a problem often makes it relatively easy to come up with a solution, or at least to begin working toward one.

Removing RPMs

I have been using Red Hat Linux for about five years, back from the days in which Red Hat was a small company in Connecticut, rather than a well-known start-up whose name even my mother recognizes. During that time, I have become a big fan of RPM, the Red Hat Package Manager. I still remember when I had to modify makefiles by hand in order to compile software downloaded from the Internet; the fact that I can download a binary version of a program, install it with a single command, and then remove it just as easily, continues to amaze me. (I'm told that Debian's packaging system is even more sophisticated, but I have not yet had the opportunity to try it out.)

System administrators who learn to love RPMs are often reluctant to begin compiling programs from source. Not only does this take more time and more knowledge, but compiling and installing a program from its source code makes removal difficult, months or years later. However, there are certain programs that I insist on compiling from source, despite the inherent disadvantages. One of these is Apache, which is normally installed as part of the Red Hat installation.

Apache itself is a relatively small program. The bulk of the functionality resides in individual modules which are normally incorporated into Apache at compile time. So if you want Apache to automatically correct misspelled URLs, you can include `mod_speling` (and no, that isn't a typo) in your compilation. If you will be running a server without any CGI programs, you can remove `mod_cgi`. And the list continues, making it possible to customize Apache according to your exact needs.

Thus, I strongly recommend that everyone compile Apache from scratch. This process is normally quite straightforward, and should take no more than a few minutes on a typical modern computer. The first step toward compiling Apache is to remove any RPMs that might already be on the system, to avoid confusing either the RPM database or yourself regarding a file's exact origins.

To remove the Apache RPM from a Red Hat system, use the following command: **`rpm -e apache`**.

To get a better idea of what is going on as your disk drive crunches away, you may turn on the "very verbose" option: **`rpm -evv apache`**.

On most systems, this command will not be sufficient by itself. RPM keeps track of not only which files are installed, but also of the dependencies between packages. Since a typical Red Hat installation includes RPMs for **`mod_perl`** (which is often broken) and **`mod_php`**, you may need to erase these as well:

```
rpm -evv apache mod_perl mod_php
```

If removing a package will break a dependency, RPM will exit with a fatal error, indicating which dependencies will not be satisfied if you remove the package in question. You should then consider whether the package (e.g., `mod_perl`) is only useful with another package (e.g., Apache) or if removing it would cause too many other problems.

Once you have removed any existing Apache RPMs, you can begin to compile and install Apache on your system. (Of course, you can always compile Apache while the RPM exists, removing the RPM just before you install the compiled version.) Download the latest version, which at the time of this writing was

1.3.12, from a local mirror of <http://www.apache.org/>. Once you have done that, you can unpack it with:

```
tar -zxvzf apache_1.3.12.tar.gz
```

The **z** option uncompresses the tar archive with **gunzip** before proceeding, and the **vv** flags ask for “very verbose” output. Both of these options work only with GNU tar, which is the standard tar version on Linux systems.

Once you have unpacked the source code, you can compile Apache with the following commands:

```
cd apache_1.3.12 # Switch into the Apache directory
./configure      # Get a default configuration
make             # Compile the source code
make install     # Install Apache under /usr/local/apache/
```

The above four steps will place the Apache-related programs in `/usr/local/apache/bin`, logfiles in `/usr/local/apache/logs`, HTML files in `/usr/local/apache/htdocs`, and CGI programs in `/usr/local/apache/cgi-bin`. The “make install” command must be performed while logged in as root.

To run Apache, use the **apachectl** program which is installed by default in `/usr/local/apache/bin`. **apachectl** is a shell script which makes it relatively easy to start or stop Apache, ensuring that only one Apache process will run on a system at a given time. To start Apache, use:

```
/usr/local/bin/apachectl start
You may need to insert this line in one of your startup files, such as
/etc/rc.d/rc.local, to ensure that Apache starts
up when your
computer is booted. This is especially true if you removed the RPM
version of Apache using the above instructions, since the RPM comes
with a startup file that is automatically placed (and invoked) inside of
/etc/rc.d/init.d. apachectl can also be used to shut down the server:
```

```
/usr/local/bin/apachectl stop
```

One of the first things that Apache does when it starts up is to look through its configuration file, traditionally called `httpd.conf` and located (by default) in `/usr/local/apache/conf`. This file contains a number of commands, known as “directives”, followed by one or more values. We can thus set the name of the server explicitly with the directive:

```
ServerName www.lerner.co.il
```

Each Apache module is allowed to define its own directives which control the program's behavior when invoking that module. For example, **mod_userdir** installs the `UserDir` directive, indicating which directory name should be treated specially inside of each user's home directory, for the purposes of creating sites on the Web.

But what happens if `mod_userdir` is not installed? Then Apache will not know what to do with the `UserDir` directive, and will exit with a fatal error. The solution is to run **apache configtest**, which reads `httpd.conf` and checks it to ensure that all of the directives are known and have legal values. If all of the directives have legal definitions, `apachectl` will respond with “OK”.

Another partial solution to this issue is to put all module-specific directives inside of an `<IfModule>` section. `<IfModule>` tells Apache that it should pay attention to one or more directives only if a particular module is actually loaded. Thus, we can always define our `UserDir` directive, assuming it is wrapped in the following section:

```
<IfModule mod_userdir.c>
    UserDir public_html
</IfModule>
```

`<IfModule>` is particularly useful when working with modules compiled as DSOs, because of the flexibility it adds.

Using Telnet

Once your Apache server is compiled, installed, configured and running, you might still encounter some problems. **apachectl configtest** is a good way to ensure that all of the directives are legal—but this does not ensure that they will work, nor that they are the values you really want.

A good tool for checking that a server is working in the right way is **telnet**. Telnet is probably familiar to most Linux users as a way to log into one computer while sitting at another. However, telnet can open a TCP connection on any port, not just the default port 23 used for the “telnet” protocol. You can thus use telnet for SMTP (port 25), POP (port 110) and even HTTP (port 80). This is a great way to test web servers to see if they're running, as well as run some basic tests.

The key to this technique is understanding that every TCP/IP service is associated with an IP address and port on both the origin and destination computers. Thus, a telnet connection between two computers will almost certainly include two IP addresses, an arbitrary port on the client, and the well-known port 23 on the server. Similarly, an SMTP connection carrying e-mail between two computers will normally include a connection from an arbitrary port on the client to port 25 on the server. For a list of well-known ports, including many that should not be changed without good reason (such as FTP, SMTP, telnet), look at the standard file `/etc/services` on any computer running Linux.

To use the **telnet** program to connect to an arbitrary port, simply indicate the port number (or name, if one appears in `/etc/services`). For example, to connect to the HTTP server running on port 80 of the computer `www.lerner.co.il`, simply say **telnet www.lerner.co.il 80**.

If the server is running on a different port, as defined by the Apache Listen and Port directives, simply specify a different number. For example, if the server is running on port 8080, we can connect to it with **telnet www.lerner.co.il 8080**.

Of course, a single Apache process can handle input on two or more port numbers, so long as they are specified correctly in `httpd.conf`.

If no server is running on the specified port, then telnet will exit with a fatal “connection refused” error. In such a case, consult the Apache error logs, normally placed in `/usr/local/apache/logs/error_log`, to determine the source of the problem.

If an HTTP server is indeed running on the specified port, then telnet will display a message indicating how to exit back into the telnet command prompt (usually with **CONTROL-J**), and will then connect you. What prompt you see depends on the type of server to which you connect; while SMTP, FTP and POP all greet an incoming user with the name of the server computer, HTTP is unusually silent. The assumption is that you know the name of the server to which you have connected, and that the connection was obviously successful.

At this point, you can issue an HTTP request. The simplest form of request is **GET /**, followed by a single newline character. This is an HTTP request of the most primitive sort, which is no longer used but is built into HTTP servers in order to ensure backward compatibility with older clients. Soon after pressing **ENTER**, you should see the contents of the root (or “index”) page of the site to which you connected. It may be difficult to read the unparsed HTML at first, but remember that this is a simple debugging method.

A more sophisticated request uses HTTP/1.0, the first version of HTTP to include headers in the request and response. The request and response are each preceded by one or more “header” lines, containing a header name, a colon and a text string. The headers and request/response body are separated by a single blank line.

Our simple request can thus be rewritten as **GET / HTTP/1.0**, with the latter part tacked on to indicate that our client understands a more sophisticated version of HTTP. We then have to press **ENTER** twice—once to indicate the end of this line, and a second time to indicate that we don't want to send any headers between the command line and the body of the HTTP request.

You can also submit name-value pairs to the server with telnet, separating each name from its corresponding value with "=", and each pair with an ampersand. Normally, such arguments are passed in a GET request to a CGI program or other dynamic content producer. For example:

```
GET /cgi-bin/foo.pl?name1=val1&name2=val2 HTTP/1.0
After pressing ENTER twice, you should see a set of response
headers, followed by whatever output is produced by the CGI program
foo.pl.
You can also pass headers along with the request. After pressing
ENTER following the initial GET line, type one or more
headers, with a new line after each one. For example:
```

```
GET /cgi-bin/foo.pl?name1=val1&name2=val2 HTTP/1.0
Accepts: text/html
Accept-language: text/html
Following the final header, press ENTER twice—once to finish the
header, and again to indicate that the request is complete.
```

How Many Servers?

Even moderately popular web sites tend to run more than one copy of Apache at a time. Older servers would wait until a new connection came in before deciding whether to "fork" off a new copy of the existing server process. The authors of Apache abandoned this method in favor of "pre-forking", meaning that Apache creates a large number of child processes immediately upon startup.

Each child process handles only one HTTP connection at a given time, meaning that there must always be at least as many Apache processes running as the number of simultaneous connections a site should handle. This maximum number is set with the MaxClients directive, which defaults to 150. If MaxClients is set too low, then one or more users may end up waiting until an Apache process finishes handling the previous connection, and is available to service a new one.

Apache dynamically changes the number of servers available depending on the number of requests that it gets, based on hints in httpd.conf. The MinSpareServers and MaxSpareServers directives tell Apache how many extra servers to keep around in preparation for incoming requests. If the number of spare servers ever goes below MinSpareServers, Apache spawns several new servers. By contrast, if there are more unused servers than defined in MaxSpareServers, Apache will kill off the extra ones.

If the server starts and responds successfully, but is taking a long time to accept connections, it could be that you have not told Apache to start enough servers. Try increasing either MaxSpareServers or MaxClients so that fewer people will have to wait for a free server to handle their request.

Of course, adding new servers is a potentially large drain on the computer's resources, consuming more CPU time and memory. **mod_perl** is a particularly large user of memory; so you should be more conservative when adding new Apache processes that include **mod_perl**. Use the standard Linux **free** command, which displays the amount of available physical and virtual memory, to get a better understanding of where the memory is going. I also like to use **top**, which displays, among other things, the amount of CPU and memory each process is consuming.

Because web servers need to respond to requests as quickly as possible, and because virtual memory is far slower than physical RAM, you should pay particularly close attention to virtual memory usage and minimize its use.

Small web sites that run one or more database servers (such as MySQL or PostgreSQL) on the same computer as a web server can find themselves in an unenviable bind. As the number of visitors to a dynamically generated site increases, the number of Apache processes must also increase. But in order to service all of these visitors, the number of database connections must also increase. At a certain point, a site becomes a victim of its own success, with the database and web site competing for system resources. For this reason, most popular database-backed sites separate the two functions onto at least two computers, with one or more database servers connected to one or more HTTP servers.

mod_status

One nice way to get a snapshot of the current Apache status is with the **mod_status** module. **mod_status** describes the current state of every HTTP server, whether it is waiting for a new connection, reading the request, handling the request or writing a response.

mod_status is compiled into Apache by default, meaning that all you need to do in order to activate it is to set the appropriate directives, and set the default handler, or request-handling subroutine, to be "server-status". Any URL defined to have a handler of "server-status" then produces a status listing, ignoring the rest of the user's request.

It is thus most common for **mod_status** to be activated for only one URL. For example, we can create the virtual "/server-status" URL on our web server, such that anyone visiting /server-status will be shown the output from **mod_status**. We also indicate that Apache should always produce a full status listing, rather than the simple version. Here is one such simple configuration:

```
<Location /server-status>  
    SetHandler server-status
```

```
</Location>
ExtendedStatus On
```

Once I put those four lines inside of httpd.conf and restart Apache—or send it a HUP signal—I get the following output from the /server-status URL:

```
Server Version: Apache/1.3.12 (UNIX) mod_perl/1.24
Server Built: Mar 29 2000 12:25:42
Current Time: Friday, 21-Jul-2000 16:02:51 IDT
Restart Time: Friday, 21-Jul-2000 16:02:48 IDT
Parent Server Generation: 2
Server uptime: 3 seconds
Total accesses: 0 - Total Traffic: 0 kB
CPU Usage: u0 s0 cu0 cs0
0 requests/sec - 0 B/second -
1 requests currently being processed, 4 idle servers
```

The status information begins with a fair amount of text indicating how long the server has been running, and how many times people have accessed the server. It also indicates just how many bytes are being served by this web process and how many servers are sitting idle. **mod_status** thus provides a nice window into the world of the Apache server, allowing us to see whether we have defined MaxSpareServers in the most resource-efficient manner.

mod_status then produces output in the following format, which can seem cryptic at first:

```
W.....
.....
.....
.....
```

Each “.” character represents a potential Apache server process which is currently not running. Those that are waiting for a new connection are represented by “_”; those that are reading input from the user's HTTP request are represented by “R”; and writing their output to the user's browser are represented by “W”. Not all letters may be visible at a given time; the current Apache status changes dynamically, and the output you see from mod_status will change to reflect that.

Following this display, we get a play-by-play view of what each active process is doing. We can see which connections are taking a long time to be processed, which connections are the most popular each month, and nearly any other facet.

Of course, it is normally a bad idea to open up your status information to the entire world. Luckily, you can use the “Order”, “Deny” and “Allow” directives to restrict access to a set of IP addresses or to an entire domain. For example:

```
<Location /server-status>
  SetHandler server-status
  Order deny,allow
  Deny from all
```

```
Allow from .lerner.co.il
</Location>
```

With the above configuration, `mod_status` will display results only for IP addresses in my domain. Requests coming from another domain will get an HTTP response indicating that access is forbidden to them.

Thinking Ahead with DSO

Until now, we have assumed our Apache installation is statically compiled, with all of the modules placed inside of Apache at compile time. This is the traditional way to compile Apache, and the default if you simply perform a “./configure”.

The problem with the above configuration is that it is relatively inflexible. What happens if you discover that you forgot to configure Apache to include a particular module? You will have to recompile the entire program, specifying which modules you do and don't want to include when running **configure**. This does not seem bad at first—after all, how often will you want to add a new module to the system?

However, the problem is deeper than that, in at least two ways. Why should Apache consume memory for modules that might not be used? In addition, why should I have to recompile Apache every time a new version of just one module is released?

In order to solve this problem, the Apache developers now support a system known as DSO, or “dynamic shared objects”. DSOs make it possible to compile Apache with only two statically linked modules, **mod_core** (which provides core functionality) and **mod_so** (which handles the loading of DSOs). All other modules can be built such that they are loaded only when necessary. Because modules are not an inherent part of Apache, they can also be upgraded without having to recompile the web server itself. As we will see below, this functionality can come in handy if you need to upgrade or debug an already-running and configured Apache server.

To compile Apache with DSOs, you will need to decide which modules you wish to compile statically, and which you will compile as DSOs. My preference is to make everything a DSO, but to compile only the modules that Apache installs by default. We can do this by changing our invocation of “configure”:

```
./configure --enable-shared=max
```

This will automatically enable **mod_so**, and will compile Apache with all of the default modules. After compiling Apache with **make** and installing it with **make install**, Apache will continue to work as before. The only difference is that it can

now load modules dynamically, adding new modules to the already-compiled HTTP server as they are needed.

The default `httpd.conf` created by an Apache server compiled with **--enable-shared** looks slightly different from that created for a statically compiled Apache server. For one, most of the directives are hidden inside `<IfModule>` sections, making it possible for Apache to load even if some of its modules have not been loaded yet. In addition, each module must first be loaded with the `LoadModule` directive, and then enabled with the `AddModule` directive. For example:

```
LoadModule perl_module libexec/libperl.so
AddModule mod_perl.c
```

`LoadModule` takes two arguments, the name of the module and the file in which the `.so` file sits. The name must match the name with which the DSO module was compiled, while the file name should point to a directory relative to `/usr/local/apache`. In the above example, and by default, DSO modules are placed in `/usr/local/apache/libexec`.

In contrast with most of the other directives in `httpd.conf`, `LoadModule` and `AddModule` are sensitive to the order in which they are placed. `LoadModule` must come before `AddModule`, and each module must be loaded and added before its directives will work. In addition, some modules must be loaded before others, and will not work otherwise. If it is possible to let an automatic configuration tool take care of the insertion of `LoadModule` and `AddModule`, let it do so. This may save your web server from hard-to-track-down configuration problems.

apxs

Once Apache is compiled with DSO support, new modules can be added at any time. However, these modules must be compiled with the same configuration information that was present when the Apache server itself was compiled. This is handled automatically by **apxs**, the “Apache Extension” program, written by Ralf S. Engelschall. **apxs** makes it possible to compile an Apache module into a DSO (`.so` file), and then to install it into the appropriate Apache configuration. Unfortunately, there seems to be little or no documentation for **apxs**, meaning that it can be difficult to understand exactly what this program does or how it works.

For example, let us assume we have already compiled and installed Apache with DSO support. Several weeks later, we notice that many of the hits on our site are resulting in “file not found” errors, because users cannot spell the odd names we have used in our URLs. One solution is obviously to revamp the site

such that users will be able to spell things more easily. But an easier solution is to install `mod_speling`, so capitalization and spelling are largely ignored.

To compile `mod_speling` as a DSO, type:

```
/usr/local/apache/bin/apxs -c mod_speling.oc
```

`apxs` will invoke `gcc`, compiling `mod_speling`. The result will not be directly executable, but rather a library that can be invoked by Apache. If the compilation was successful, then we can install `mod_speling` with the following command:

```
/usr/local/apache/bin/apxs -i -n -a mod_speling.so
```

Using `apachectl configtest` is particularly useful when installing new DSO modules. It ensures that the module we have added is indeed there and working, and that Apache now understands any new directives we added outside of `<IfModule>` sections.

`mod_perl`

`mod_perl`, the Apache module which makes it possible to write new modules in Perl and to configure existing ones using the popular language, can be compiled as a DSO. This requires the use of `apxs`, as in the case of `mod_speling`. However, `mod_perl` is much more complicated than a single module, and depends on many outside pieces of information for its compilation. `mod_perl` is thus configured and compiled similarly to stand-alone Perl modules, with `perl Makefile.PL`, followed by a `make`, `make test` and `make install`.

In order to compile a new version of `mod_perl` into an existing version of Apache, issue the following command:

```
perl Makefile.PL \  
USE_APXS=1 WITH_APXS=/usr/local/apache/bin/apxs
```

If you want to enable `mod_perl` for all of the different Apache handlers, rather than just the default `PerlHandler` (for creating dynamic content), turn on the **EVERYTHING** switch. For example:

```
perl Makefile.PL \  
USE_APXS=1 WITH_APXS=/usr/local/apache/bin/apxs \  
EVERYTHING=1
```

Once you have created the Makefile in this way, you can compile and install `mod_perl` into the existing Apache server with:

```
make  
make test  
make install
```


This method works not only for installing a new copy of mod_perl into Apache, but also for upgrading an existing copy. You can then test to see if mod_perl has been compiled into the server by telneting to the server's address and port number, and issuing the command:

```
HEAD / HTTP/1.0
This will return the HTTP headers associated with the / document on
the server. Among other things, there should be a "Server" header
indicating what kind of server is running. mod_perl adds a tag to
this output string, so you should see output like the following:
```

```
Server: Apache/1.3.12 (UNIX) mod_perl/1.24
```

Because mod_perl updates come out at different times than Apache updates, I have found it to be extremely useful to install and upgrade mod_perl in this way.

Apache::Status

While mod_status can tell us what is happening with each Apache process, it cannot tell us what is happening inside a particular module. In general, this is not such a bad thing; do I really care what is happening inside mod_mime or mod_speling?

But in the case of mod_perl, where so many complex things are going on, it would be nice to be able to find out what is going on. Perl Apache::Status module, which works with mod_perl, provides this information.

In order to activate Apache::Status, we need to insert another section into httpd.conf. As with mod_status, we will create a new **Location** section which associates a particular handler with a virtual URL, traditionally known as "/perl-status":

```
PerlModule Apache::Status
<Location /perl-status>
    SetHandler perl-script
    PerlHandler Apache::Status
</Location>
```

Once we have restarted the server or sent it a HUP signal, requesting the URL /perl-status will produce a menu of options, such as "Environment" and "Inheritance tree". Other Perl modules for mod_perl, such as **HTML::Mason**, can install their own hooks for **Apache::Status**, making it possible to look through their environments. For example, Mason provides a simple interface for viewing the current configuration, as well as a list of components that have been compiled and cached.

A Short Story

Over the days preceding my writing of this column, I used all of the above techniques to track down a problem with my installation of **HTML::Mason**. The colocated server that I help run was having some problems handling a growing load. Every few hours, all of the Mason-based sites on the server would fail, followed one or two hours later by the non-Mason sites. Actually, “fail” is too strong a word—the browser would send a request to the server, but would time out (after ten minutes or so) of waiting to receive a connection. What was going on, and how was I going to fix it?

My first reaction was to think that our server was running out of RAM. I used “top” and “free” to inspect the system, but did not see anything out of the ordinary. On the one hand, this was a relief. At the same time, this meant we were somehow running out of available servers, even though I had configured the system for a maximum of 150 simultaneous clients. My server might be popular, but 150 simultaneous accesses is highly unusual, even for me. Something else was obviously going on here.

There was clearly some connection to Mason, and perhaps to mod_perl. I decided it was about time to upgrade mod_perl to the latest version (1.24), using the technique I described above. So I upgraded the copy of HTML::Mason and several other related modules, restarted Apache with apachectl, and hoped the problem would go away.

Unfortunately, the simple upgrade did not do the trick. The system still stopped responding to requests after several hours of activity. I used Apache::Status to look at the state of mod_perl, and nothing seemed to be out of the ordinary.

I looked at the system status with mod_status, and discovered that over time, many of the Apache processes were getting stuck in the “write” state. In other words, the list of Apache processes was slowly, but surely, being transformed from a list of “.” (unallocated) processes to a list of “W” (writing an HTTP response) processes. Every invocation of a Mason component on the system was using up another Apache process, and never letting it go! It was no surprise, then, that the system was locking up after only a few hours; if the Mason-related parts of the site had been more popular, the system would have gone down even more quickly.

I looked through the Mason configuration file and determined that my use of the Apache::Session module, which allows mod_perl programs to track a user's movements and actions (as we saw in my article, “Session Management with Mason” in the August *Linux Journal*) was failing to return. So each time a Mason component was invoked, everything would work fine—until the component

needed to return, at which point the executing program would simply spin its wheels, waiting to connect to the MySQL database.

My solution was not particularly elegant, but did the trick: I decided to stop using the MySQL version of Apache::Session (known as Apache::Session::MySQL) and start using the simple file-based version (known as Apache::Session::File). I restarted the server, and was delighted to discover that everything was working as it should.

Conclusion

Apache is a wonderful, robust and easily configurable HTTP server. However, it is also a complex piece of software that requires some experience in order to tune correctly. When mod_perl is added to the mix, the complexity increases even more. Luckily, it is possible to install Apache such that installing and upgrading modules can be easy and painless.

Moreover, a number of modules and tools (such as mod_status and Apache::Status) make it possible to view the internals of Apache as it is executing. By taking advantage of these tools, we can find and fix problems with our servers, spending our time on more interesting issues than trying to figure out why a server does not run.



Reuven M. Lerner owns a consulting firm specializing in web and Internet technologies, based in Modi'in, Israel. As you read this, he should (finally) be done writing *Core Perl*, published by Prentice-Hall. You can reach him via e-mail 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.

Cooking with Linux: A Few Recipes for Easier Firewalls

Marcel Gagné

Issue #78, October 2000

Simple ways to secure your system.

François! You call that a security notification? The message must contain more information than "This is a test of the emergency security broadcast system", if any self-respecting system administrator is going to be able to act on it. *Qu'est-ce que je vais faire avec toi?* François, what are you looking at? What? Oh, *mes amis*, forgive me. I did not notice your arrival. We were setting up some new security policies on our Linux servers, and François had written a script to detect my port scans and I was completely distracted.

François! Wine for our friends. *Vite! Vite!* Come, *mes amis*. Let me show you to your tables. As you get comfortable, I should tell you about this message my waiter had the system send out. I did a thorough port scan on the system, and all he sends as notification is that old "This is a test of the emergency et cetera" type of message. Not much of a description, *non?* You know, when I was younger and they had those messages on television, I always suspected that if there had been a real emergency, we would not have gotten the message because those who knew of the problem would already have been in hiding. A little joke, *mes amis*. Ah, François. Yes. An excellent choice of wine. You will like this one, *mes amis*, a superb 1995 Montrachet. Please pour, François.

Security, as you know, is very serious business indeed. Every day, we hear of damage caused by viruses, of new exploits through which crackers compromise systems. For those of us in the information technology restaurant business, these are challenging times. We must be ever vigilant. A good firewall, then, is an excellent beginning. But how to do it simply is the question, *non?*

The simplest way to secure your machine (short of locking it up in the wine cellar with no Internet connection) is to disable all nonessential services from your `/etc/inetd.conf` file and let your TCP wrapper program control who gets in

and who stays out. The three lines in Listing 1 are samples from an `/etc/inetd.conf` file.

Samples from an `/etc/inetd.conf` File

Listing 1. Samples from an `/etc/inetd.conf` File

Notice that the third line is commented out. That service (rlogin) is simply not available on this system, while telnet and FTP are. Now, the `/usr/sbin/tcpd` you see is actually the path to the TCP wrapper program. It decides whether those services that are uncommented will be allowed as incoming traffic. If someone tries to access services that are denied, you'll get a log of that in your `/var/log/secure` file with a nice "refused" message to let you know someone was being bad. Here's how it is done.

Add the following line to your `/etc/hosts.deny` file:

```
ALL:ALL
```

You start by denying all access by anybody. The first ALL refers to all services. The second ALL simply means everyone. You probably expected there would be a `hosts.allow` to counter the `hosts.deny` file, and you are correct. That file contains the hosts you want excluded from this global refusal of service. Let's assume your internal network has a base address of **192.168.1.0**. The hosts we want to be able to telnet or ftp to are the ones in that network. In the `/etc/hosts.allow` file, you would put these entries:

```
ALL: 127.0.0.1<  
>  
ALL: 192.168.1.
```

The dot after the **1** means that **192.168.1.anything** is acceptable. The last thing to do is restart the master network process, `inetd`. On most Linux systems, it can be done like this:

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

As I mentioned, this is pretty simple, but also covers some pretty basic ground. Perhaps *too* basic. For instance, your TCP wrappers cover only those services listed in `/etc/inetd.conf`. A better way to build a firewall is through the use of `ipchains`, a packet filtering system for Linux machines running kernels starting just before version 2.2. The `ipchains` program is the successor to the older `ipfwadm` program. Most newer distributions will require `ipchains` for packet filtering.

If you have read the `ipchains` documentation, you've probably also felt a twinge of panic as you started to realize how complex this can all be. I'll spend the next

few paragraphs with some (I hope) simple explanations of the process, then I will show you a few recipes from some great open-source chefs that will make the process less frightening and more fun.

Ipchains is a command-line utility that lets you create packet filtering rule sets called "chains". These chains come in a few different flavors: **input** (packets coming in from the outside), **output** (packets bound for the outside world) and **forward** (packets being routed through your system, as in the case of ip forwarding and masquerading). There is actually a fourth, which can be user-defined and named. The format of the command is similar to the following:

```
/sbin/ipchains -A forward -j MASQ<
>
-s 192.168.1.0/24 -d 0.0.0.0/0
```

The **-A** means to add a rule to the chain. Other options are **-D** to delete, **-R** to replace, **-I** to insert, **-N** to create a new user-defined chain, and a handful of others. The above command creates a rule set that will forward packets from any host on my internal network and masquerade them to appear as though all traffic was originating from one machine only. The **0.0.0.0** means the packet destinations can be anywhere. The **-j** flag defines the action for this rule. Other than the above, which will do masquerading for your site, you can also specify **ACCEPT** (let the packet through), **REJECT** (reject the packet, but let the other side know it is being rejected) and **DENY** (don't allow the packet through and don't offer any explanation).

What else can we do with this? For one thing, I could create a pretty safe system by adding this single rule:

```
/sbin/ipchains -A input -j DENY -s 0.0.0.0/0 -d<
>
0.0.0.0/0
No one will get into your system, and no one will ever know why. It might
be an idea, then, to insert this rule into the chain before you
do the above. Then at least your local network will have access to
your server.
```

```
/sbin/ipchains -I input -j ACCEPT -s<
>
192.168.22.0/24 -d 0.0.0.0/0
```

Let's try something a bit more complex. Starting from scratch, I might set up the following input chain. Note that there should be no broken lines here. Each command in the list begins with **/sbin/ipchains**. You will probably also notice that my Internet address is an imaginary one. No IP address starts with 259.

```
#<
>
# Input Rules
# /sbin/ipchains -F input
# /sbin/ipchains -P
input ACCEPT
# /sbin/ipchains -A input -j ACCEPT -s
```

```

192.168.1.0/24 -d<
> 0.0.0.0/0
# /sbin/ipchains -A input -j ACCEPT -s
0.0.0.0/0 -d 0.0.0.0/0
# /sbin/ipchains -A input -j DENY -p tcp -s
0.0.0.0/0 -d 259.25.132.55<
> 137:139
# /sbin/ipchains -A input -j DENY -p udp -s
0.0.0.0/0 -d<
> 259.25.132.55 137:139
# /sbin/ipchains -A input -j
ACCEPT -p tcp -s<
> 0.0.0.0/0 -d 259.25.132.55 80
# /sbin/ipchains -A
-d 0.0.0.0/0

```

We start by “flushing” our input chain. That's the **-F** option at work. We then assign a default policy of **ACCEPT** with the **-P** option. The next two rules allow all traffic from our local network and the outside world to enter. Next, we **DENY** all traffic bound for the netbios services (Windows or SAMBA file sharing) at ports 137 through 139. A range of TCP ports is expressed by a starting number, a colon and an ending number. Since we do want to allow web server access on this mythical system, we then open port 80. Finally, we catch anything that is still open and does not fit the above rules by closing off access as earlier with an all-inclusive **DENY**.

Mais oui, the security-conscious gentleman at table twelve questions my reasons behind starting out with a policy of **ACCEPT** instead of **DENY**. You are wondering, has Marcel been sipping too much of his own wine? Well, *mes amis*, when constructing this example, I considered the creation of a rules set, added line by line remotely, whether from your own internal network, or from outside. If you **DENY** everything immediately, you deny yourself as well. Now, when it comes time to script all these commands, you may want to consider setting up a default **DENY** policy rather than **ACCEPT**. Even so, when the rules are executed in a script at boot time, with the final result being that you **DENY** anything that has not been specifically ACCEPTed, the window for a cracker to exploit this policy is very small.

At any point, you can list all the rules by typing **/sbin/ipchains -L**. You can also flush the table of all chains, rule sets, etc. by typing **/sbin/ipchains -F**. This is by no means a complete firewall setup, nor is it entirely realistic. In a real setup on a real server, there are other ports I would like open (such as e-mail or DNS services). I would also like to explicitly close access to my X window sockets (ports 6000, 6001, etc.).

If the prospect of creating a firewall from scratch seems a bit daunting and you would prefer a very quick-and-dirty approach to a rules-based IP filtering system, look no further than the “ipchains-firewall” script distributed by Ian Hall-Beyer. To create a basic firewall, you need only call the script with your outside interface and inside interface as the parameters. Here's how I did it on my system with an external PPP connection:

```
./firewall.sh ppp0 eth0
```

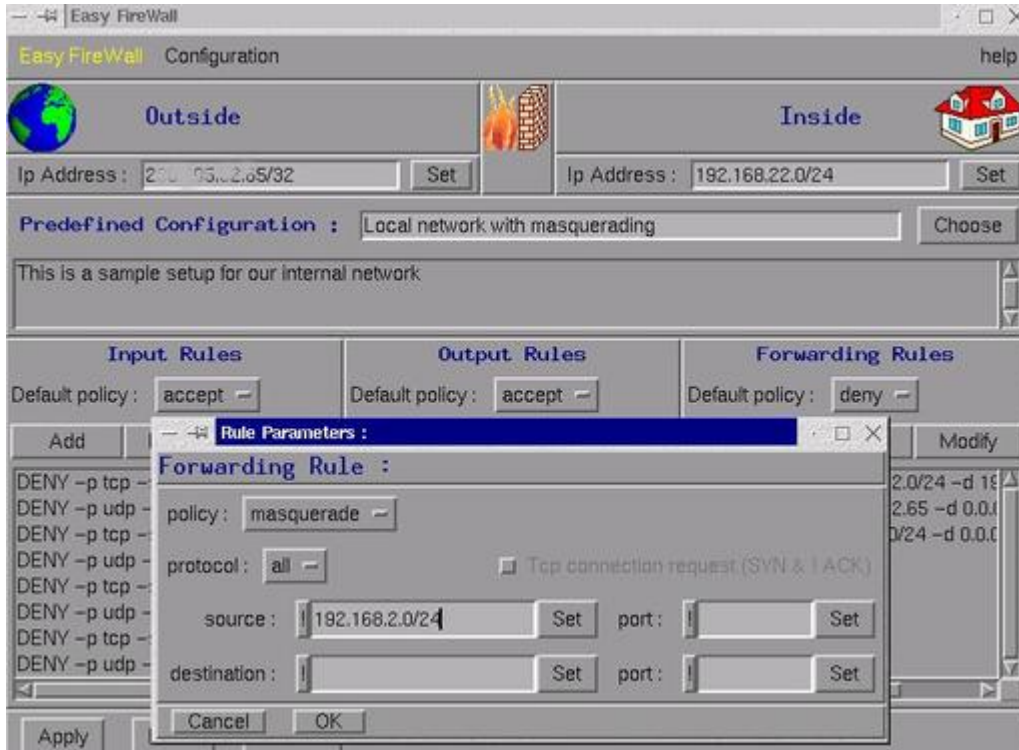


Figure 1. Running the firewall.sh Script

As the script runs, it will set up masquerading, block out remote access to things like netbios (SAMBA and Windows file sharing) and X sessions. You can see the script in action in Figure 1. Using `/sbin/ipchains -L`, you can list the rules created by the script. Ian's script is the beginnings of a firewall system, taking into consideration the "usual". Customization can be done just by editing the script, which you will no doubt want to do. The defaults are fairly restrictive. For instance, vital services like SMTP are, by default, denied. The following example comes from the script itself, showing where I have uncommented port 25 (among other things):

```
# telnet (23)<
>
# $IPCHAINS -A input -p tcp -s 0/0 -d $1 23 -j
ACCEPT
# echo -n "."
# smtp (25)
$IPCHAINS -A input -p tcp -s 0/0 -d $1 25 -j
ACCEPT
echo -n "."
# DNS (53)
$IPCHAINS -A input -p tcp -s 0/0 -d $1 53 -j
ACCEPT
$IPCHAINS -A input -p udp -s 0/0 -d $1 53 -j
ACCEPT
echo -n "..."
```

What? Ah, you are asking if we have anything with a bit more spice on the menu. *Mais oui*. For those who prefer a graphical approach, I would like to

recommend the following two items, Easy Firewall and GTK+ Firewall Control Center.

The brainchild of Daniel Roche, Easy Firewall is a Tcl/Tk (version 8) application that provides a nice simple interface for firewall administration. For those out there running older Linuxes, this application also works with **ipfwadm**. The program is available in a tarred and gzipped bundle or a Red Hat RPM. Since it is a Tcl/Tk application, it's also very easy to work through the program in case you happen to be curious. Installation is easy in either case, since there is no compiling to do with the Tcl/Tk script.



```
mgagne@nexus.salmar.com: /home/mgagne
[ root@website ipchains-firewall-1.7 ]# ./firewall.sh ppp0 eth0
Checking External Interface...found ppp0
External Interface Data:
Address: 0.203.02.05/255.255.255.255
Network: /255.255.255.255
Checking Internal Interface...found eth0
Internal Interface Data:
Address: 192.168.22.10/255.255.255.0
Network: 192.168.22.0/255.255.255.0
Checking internal interface for RFC1918Class C found
Going to Masq Mode
Flushing rulesets.....Done!
Outbound connections....Done!
Setting Ipchains timeout for tcp tcpfin udp.. Done!
Masquerading.....TOS flags.....Done!
Loopback...Done!
Cable Modem Nets...Done!
Port Blocks.....Done!
High Ports...Done!
Services....Done!
ICMP Rules.....Done!
[ root@website ipchains-firewall-1.7 ]#
```

Figure 2. An EasyFw session

Easy Firewall comes with a few pre-defined firewall configurations that are easily modified. Start the program with the command **easyfw**. The big friendly screen lets you choose your internal and external interfaces at the click of a button. It will then scan your system for any ipchains rules that may already be in effect, and display them in the interface. Modify, add or delete rules as you see fit. When you have a satisfactory configuration, you can choose to **Apply** that configuration to your running network (now or at boot time) as well as save to a file. The file option is great for a glimpse into the structure of the firewall rules. Furthermore, since that file is a shell script, you can call it from your /etc/rc.d/rc.local at boot time.

Our next offering comes from Koo Kyoseon, a little something called the GTK+ Firewall Control Center. In the style of our previous chef, Koo has built a clean, friendly interface to ipchains administration. This program is available in either RPM or source code format. If you will be building it from source, note that you will also need the libipfwc library. Luckily, this is included with the source for

gfcc. At the time of this writing, I picked up version 0.7.4 of the package, **gfcc-0.7.4.tar.gz**. Here are the steps for building the package and the library:

```
tar -xzvf gcc-0.7.4.tar.gz<
>
cd gcc-0.7.4
```

Once inside this directory, we can build **libipfwc**:

```
tar -xzvf libipfwc.tar.gz<
>
cd libipfwc
make
```

Now, change back to the gcc directory and build it:

```
cd ..<
>
./configure --with-ipfwc=./libipfwc
make
make install
```

Start the program by typing **gfcc**. This program actually has two different ways of saving information. One is a list of rules very much like what you will see if you look at the existing rules in the **/proc** file system. You can do this by typing this command:

```
cat /proc/net/ip_fwchains
```

If you want a nice shell script like the one I mentioned earlier, you can use the **export** function to create the script. For instance, if I export my rules to **/usr/local/Admin/fwrules.sh**, I can load it the next time I reboot by including that path as a single-line command at the end of my **/etc/rc.d/rc.local** script.

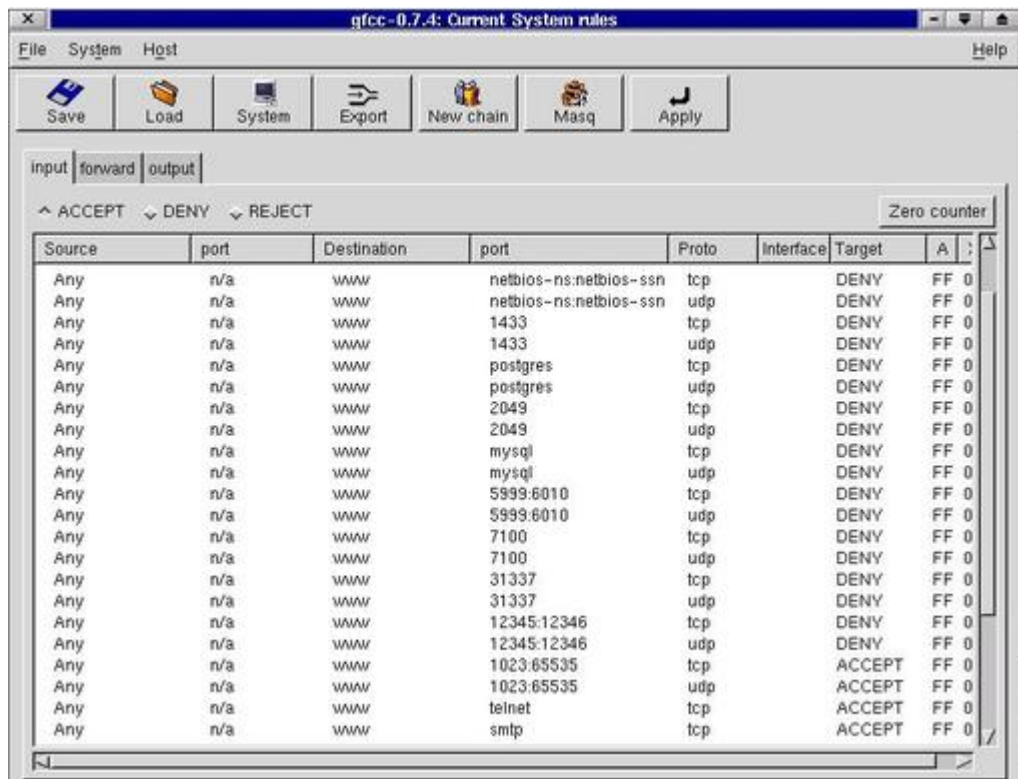


Figure 3. Configuring a Firewall the gfwc Way

Now that we have all these ideas on the table, let me make an alternative serving suggestion. If you are really new to this and you feel uncomfortable embarking on building your own firewall, use Ian's **ipchains-firewall** script to generate the initial set of rules. Then, use either **EasyFw** or **gfwc** to modify the rules. It's an easy way to get started on your way to a more secure system.

Alas, the clock, she is telling us that closing time is near, *non*? Before you go, François will refill your glasses a final time. Security is a vitally important subject. Linux's powerful, network-ready architecture opens many doors, including some you may prefer leaving closed. With a little experimentation in the Linux kitchen, locking those doors need not be a frightening experience. Until next time, I invite you to network safely, and please, lock the door before you leave. We'll see you next time at Chez Marcel. Your table will be waiting.

A votre santé! Bon appétit!

Resources



Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy, and edits TransVersions, a science fiction, fantasy, and horror magazine (soon to be an 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. You can discover lots of other things from his web site at www.salmar.com.

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

[Advanced search](#)

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

[Advanced search](#)

Linux Drives Digital Audio Revolution

Linley Gwennap

Issue #78, October 2000

Home music networks of the (near) future will bring unlimited choice into your living room.

Can MP3 really send CDs to the same dusty bin as vinyl records and cassettes? Not if the only way you can listen to tunes is by using your PC or a battery-powered player that holds an hour of music. But fear not, digital denizens—a number of new devices are emerging to MP3-ize everything from stereo components to boom boxes. And Linux will play a key role in this potentially huge new market.

The New Media: No Media

For too long, digital music has been tied to a specific medium: the 120mm CD. To listen to a song, you carry a disc from player to player. That isn't so bad with one disc, but carrying an entire collection around is impractical. And until the recent availability of CD burners, there was no way to create new compilations of songs you like or get rid of ones you don't. But digital music is just a bunch of bits. The combination of new compression techniques such as MP3, with faster modems and bigger hard disks, has made it practical to move digitized songs from one device to another. It has also become trivial to create (legally or illegally) new copies of songs.

Today, this process is typically performed on a PC, which is the only device that most people own with a network connection and a hard disk. But a PC is not ideal for this task. It takes a long time to boot, the user interface is complex, and it crashes frequently. In addition, PCs are too expensive to put in every room, and they often don't have high-quality sound outputs.

The solution is to create a home music network. Start with a music server. This stand-alone device has a hard disk, a network connection, a small screen or TV connection for the user interface, high-quality audio outputs and inputs, and

probably a CD drive. Users can download music from the Internet and also “rip” their own files from the CD drive or external audio sources. The user can listen to stored music using playlists or listen to streaming Internet radio.

The natural operating system for this device is Linux. No Windows compatibility is needed in this appliance, but Linux provides out-of-the-box networking and enough power to handle downloading, ripping, streaming and listening at the same time.

Now add music clients to the home network. These low-cost devices send requests to the music server and play streaming audio from the server or from the Internet. No hard disk or CD drive is required, but Linux is again a good choice, given its low cost and network support. Just as your home may have several radios and CD players, future homes will have several music devices connected to the music server.

A number of technologies, by the way, are vying to become the home network of choice, including phone line (HPNA), power line (HomePlug) and wireless (802.11). Any of these can support a home music network of several devices. With this new way of listening to music, there is no physical medium. Songs are accessible as bit streams from any device that is hooked to the network. In theory, you could listen to your music from a hotel room in Iowa, assuming it had an Internet connection. You could also download songs to a variety of portable devices that could each hold some or all of your music collection, allowing you to listen to your music when not connected to the Net.

Devices Beginning to Emerge

The first company to get behind this vision was S3, which owns the Rio line of portable MP3 players. The company recently announced the Rio Receiver (<http://www.riohome.com/>), which is a music client that uses the PC as a server, connecting through your existing phone wiring (without disrupting your phone service). The device sells for about \$250 US, although you will also need to add a \$49 phone-line networking card to your PC if you don't already have one. S3 is developing a music server, but that device is not yet available. Both S3 and partner Dell Computer are selling the new devices.



Figure 1. The Rio Receiver

Other small companies including Lansonix, Lydstrom and Zapmedia are marketing digital music systems with various levels of networking capability, but only S3 is producing both music servers and low-cost music clients. Its Rio was the first and is still the leading portable MP3 player, and S3 is leading the way with home stereo components as well. While only a third of all Americans regularly use the Internet, almost everyone listens to music. Digital music, in MP3 and other formats, could be the killer application that drives the adoption of home networks. As the easiest way to get a non-PC device onto a network, Linux will have many opportunities in this new market.



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 #78, October 2000

[sgalaAlive](#), [Build Your Linux Disk](#) and more.

Sometime ago, I followed a thread on the Linux kernel mailing list that talked about Linux on the desktop. Of course, people discussed why Linux had a long way to go on the desktop (no one thought it was ready). I didn't jump in with both feet, since I thought the thread was off-topic to begin with for the already-busy kernel list, but I must disagree. The focus of the conversation was on a lack of certain things like applications and desktop simplicity.

Well, I converted an entire school computer lab (33 computers) to Linux just recently, and it looks like there is more to come. I've found that, properly installed and configured, Linux on the desktop is easier for students and teachers to use (and a lot less frustrating) than anything they've used before. The secret is simplicity, since every app I can think of except Quicken has a viable alternative. While I installed Caldera (the distribution is really unimportant), the desktop is not KDE (Caldera's default). I had to put together a couple of simple shell programs attached to icons for the nontechnically inclined teachers, but nothing difficult, and the students haven't found the games (yet). They're so happy with it, they're talking about converting everything, not just this one lab.

So don't underestimate Linux on the desktop. I've found that once it's installed for the technologically challenged user, they're much happier than with any other system they've used. And those users couldn't install a Microsoft OS anyway; they're just using what comes with the system—all I provided was a viable alternative. Not bad for a guerrilla operating system.

[sgalaAlive](http://www.sgala.com/): <http://www.sgala.com/>

Have some processes you need to keep running? I know it's no fun to find your web or ftp server down. This Perl script can take care of making sure any

essential services stay up, and notify you by e-mail when they are restarted. Makes it very simple to keep track of when services go down. And this particular program is almost too easy to install and configure. It requires Perl, cron (suggested).

Build Your Linux Disk: <http://byld.sourceforge.net/>

You've probably seen a number of boot/root disks out there that can be used as rescue disks. You may even have one. But, have you ever wanted one with your favorite utilities on it, not someone else's favorites? Then you've got to roll your own. BYLD, Build Your Linux Disk, can help you do just that. And in case everything won't quite fit on one disk, the author includes information on making your disk slightly larger. In fact, a 1680K disk is standard in the config file (just look in /dev for some common sizes, like 1760 or even 1840). It requires bash, a number of common system functions, kernel sources.

Checkout: <http://www.draenor.org/checkout/>

A very simple checkin-checkout system for a company. Supports "views" based on departments. You can tell quickly where someone is (in/out) and the reason (if they've bothered to say). Will require some small tweaks to the HTML code, unless you're satisfied with the very plain interface provided. It requires a web browser, web server with PHP and MySQL support, MySQL.

Pronto Mail: <http://www.muhi.net/pronto/>

Pronto Mail is a spinoff of the GTK-based mail client CSCMail, reviewed here a few months ago. The CSCMail authors have decided to move to C, so some users of CSCMail who like Perl have moved the Perl version along. One good aspect of Pronto is the simple install. If you want to use anything but CSV (MySQL, PostgreSQL, mSQL, etc.), you'll need to create a blank database with the user's name (and permissions for the user). Other than that, Pronto is self-installing, including downloading. Good-looking and stable. It requires Perl, Perl modules: Gtk::XmHTML, Date::Manip, DBI, Text::CSV_XS, SQL::Statement, DBD::CSV, MIME::Base64, HTML::Parser, IO::Wrap, MIME::Parser, Mail::Header, MIME::Types, URI::URL, IO::Socket, Lingua::IsPELL.

passwdgen: members.home.com/denisl/passwdgen

I've used several password-generation programs. Until recently, I used makepasswd. But passwdgen has a few switches that allow you to decide if you want passwords to be all uppercase, all lowercase, all numbers, or printable characters and not numbers or letters. You can mix and match the switches to generate a remarkably difficult password. You can also specify passwords that

can be typed with just the left or right hand (obviously this works only on a standard QWERTY keyboard). It requires glibc.

Simple Network TOP: <http://sntop.sourceforge.net/>

Do you have a large number of hosts to monitor? Need to know only if they're up or down (not whether a particular service is running)? This utility is quick and easy to set up. It also has a secure mode that allows it to run on a console with the command keys disabled. Or, if you prefer HTML output, there's a switch to provide it. sntop uses fping to probe, so it won't put much of a load on your system or the network. It requires ncurses, glibc.

Subnetwork Explorer: <http://cyst.org/>

Most of us should be familiar with nmap, but it's a bit heavy just to keep an eye out for a quick check of "is someone running a rogue ftp site on the network?" type scan. Netexplorer gives you a way to make a quick check of a very few servers, currently anonymous FTP, SUNRPC and NetBios file sharing, although more will certainly be added in the future. A quick netexplorer of a class C subnet takes only seconds, as compared to several minutes for nmap, which you really should use if your needs are for a detailed network scan. It requires glibc.



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

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

[Advanced search](#)

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

[Advanced search](#)

Linux Opens Up Your TV

Rick Lehrbaum

Issue #78, October 2000

Indrema's revolutionary high-end game console will capitalize on embedded Linux and open-source software development.

Indrema's web site (<http://www.indrema.com/>) proclaims, "The future of Linux is on your TV." Reading on, I was soon informed that the Indrema entertainment system would be a "revolutionary product built on a revolutionary operating system, Linux". An easy-to-use entertainment appliance: "Just turn it on, and sit back on your couch."

Not a PC, mind you. "This is no desktop Linux system," continued the website pitch, "this is Linux for TV, for game addicts, for total entertainment. Out of the box, you can hook it up and begin playing unbelievably realistic 3-D games, browsing the Net at high speed, or just enjoying personal TV or MP3 favorites."

"This is not your normal consumer electronics company," I thought. Indrema clearly seemed to have wrapped itself around Linux—in a big way. Questions like "Who are these guys?", "What are they up to?" and "What's their angle with respect to open source?" were coursing through my brain cells.

Desiring answers to these and many other questions, I quickly phoned up Indrema's CEO, John Gildred to request an opportunity to chat. A week later, Gildred and I spent an intense hour exploring his new start-up's dreams, strategies, product ideas and open-source philosophy. Here's what I learned.

Who is Indrema?

"It was sort of a brainchild that my partners and I had, while playing Quake late at night," said Gildred, looking back at the genesis of his year-and-a-half-old start-up. "We were thinking: wouldn't it be great if somebody created an open-source game platform so that a guy like John Carmack (of Quake fame), or the

next great developer of a new gaming paradigm, could get it to market a lot quicker and could get into the [game] 'console space' a lot easier?"

Gildred and the other Indrema founders observed that there were lots of innovations taking place for PC-based games, but not much for consumer game consoles, due to high barriers to entry for individual developers, which kept them from breaking into the console market. So, they resolved to create a new game console. One designed from the ground up, to provide a game development environment and infrastructure capable of enabling any level of developer—from an individual to a large corporation—to bring products to market easily, and without huge barriers to entry. And they decided that the keys to accomplishing this mission would be open-source software, open APIs and the Linux operating system.

The concept quickly gathered momentum. Developer interest was high. Best of all, the required technologies appeared to be available.

What Will Indrema's Product Be?

The Indrema Entertainment System (IES) is packaged in a sleek enclosure with the look and feel of a top-of-the-line VCR.



The Indrema Entertainment System

"You don't know that it has Linux in it," said Gildred. "You turn it on, and it runs like a consumer electronics device. You can watch TV as you normally would. But you can also pull up a screen and start playing your MP3s. Or start the web browser and begin browsing on the Internet."

The device offers a choice of broadband access, via its built-in 10/100-megabit Ethernet interface, or a dial-up connection. It comes with a game controller and will have at least one pre-loaded game so the user can begin playing right away. There are two ways to add games: by using the built-in DVD drive, or by downloading games purchased on-line.

"Isn't it, basically, a multifunction set-top box?" I naïvely asked.

“Set-top box,” replied Gildred, “is a term that we run away from!” Gildred went on to explain that Indrema is determined to avoid having the IES positioned as a set-top box. “First and foremost, it's a game console,” emphasized Gildred. “What the IES does best is play games. It plays games very well, is extremely fast, and offers an open development environment.”

Nonetheless, that's not all the IES will do. After all, it's a fully functional multimedia Linux computer. “You will see applications that are designed for an IES platform that go beyond gaming,” added Gildred, “because it has a lot under the hood that allows additional audio/video capability.”

Some likely possibilities, mentioned by Gildred, include providing enhanced HDTV capabilities and downloading and playing music, video, and TV from content partner sites. Oh yes, and “Personal TV”. Users will be able to download and play specific TV programs on demand. Not all IES capabilities will be available with the entry-level game console system. Personal TV, for example, will be reserved as a high-end (extra-cost) option.

What's in the Box?

Let's take a look at what's inside that sleek Indrema box. The core computer consists of a 600MHz x86-compatible processor combined with a dedicated, and relatively customized, graphics pipeline. The graphics subsystem includes MP3 and AC3 encoder/decoders, digital-to-analog converters, and a specialized graphics processing unit (GPU) made by NVIDIA.

Memory is fixed at 64MB, but the hard drive can vary from 8 to 50GB depending on the console model. Of course, there's a 100Mbps Ethernet port for fast Internet connection, plus a slew of USB ports (4, in the baseline model) making game controllers—and other devices—a snap to connect.

TV outputs will drive standard composite video, S-Video and component HDTV. Inputs for composite video and S-Video are also provided. On the sound side, there are stereo analog audio outputs and inputs, plus a digital audio output port.

What CPU will Indrema use in the IES? That hasn't been finalized yet, as far as the production models are concerned. Gildred says it's going to be an as-yet-unannounced next generation Intel or AMD processor—“something new, very fast, and really catered to what we're trying to do.”

The system's internal electronic circuitry is modular in just one respect: there is a slot (on the rear of the unit) for a user-replaceable “GPU card” that houses the NVIDIA GPU and associated video frame-buffer memory. Corresponding to this

modularity, Indrema has abstracted the GPU functionality in a software driver, so that the system can adapt to future plug-in GPU upgrades.

The four USB ports provide the main means for hardware expansion. Since the IES represents a fully functional multi-media, Internet-connected Linux computer, it's no surprise that Gildred expects those ports to accommodate a wide assortment of interfaces—beyond just game controllers. High-end models will offer even more USB ports on their rear panels.

What Makes It Tick?

What makes the IES tick is, of course, its software. Basically, there are three layers of software involved:

- an open-source Linux-based operating system, called “DV Linux”
- proprietary software components that are unique to the IES platform hardware (these are distributed in binary form only, and not open-source)
- various application programs and games

A free software development kit (SDK) will be distributed on-line at the Indrema developer web site. That SDK will include OpenGL, OpenAL, OpenStream and Extrema. The first three are open source while Extrema, which is one of Indrema's proprietary software components, will be offered in binary form only.

Why Open Source? Why Linux?

Despite the fact that Indrema intends to keep certain software modules proprietary, the company is plainly a strong advocate of open-source software. Indrema participates in several open-source projects—including the Linux kernel, OpenAL and Mesa 3-D (the open-source implementation of OpenGL). In addition, the company is pioneering a new open-source streaming video architecture called OpenStream.

One of the most important Indrema projects has been to create DV Linux, an open-source Linux distribution targeted at devices using TVs for display. “We want DV Linux to be the standard, and we want people to realize that DV Linux is truly open source,” explained Gildred. “That allows us to standardize on a game platform so we can get a distribution [pipeline] going for all the game developers. We're taking steps to be sure that this is something that is supported by several players, not just Indrema. DV Linux is a tool that enables the IES platform, including our development environment, to be very open.”

Production Plans

Although earlier announcements indicated a target delivery date of the 2000 end-of-year holiday season, the current plan is to begin shipment next spring. The target retail price for the entry-level IES is rumored to be \$299. Developers won't have to wait as long. Indrema expects the free game development SDK to be available for download by game developers this fall.



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.

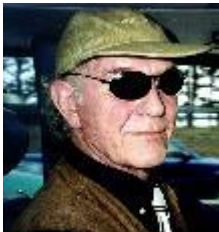
[Advanced search](#)

All Hands to the Sieve

Stan Kelly-Bootle

Issue #78, October 2000

A recent FLASH alert from SANS reveals the dangers lurking in the bowels of MS Access, Word 2000 and Internet Explorer 5.00+.



If you want to be thoroughly depressed by the CS “state of the art”, apart from running/crashing Windows, you should check out Peter Neumann's RISKS reports and the regular lists of “vulnerabilities” issued by the SANS Institute. No OS, kernel, compiler, library, parser or command seems immune, however “mature”. *Mon dieu*, we are not just talking kilobillion (who knows?) spreadsheet flaws or megabillion lost spacecraft (good for some), but pandemic, apodictic, uncaught exceptions based, in spite of endless warnings, on the syntactic quirks of C arrays.

A recent FLASH alert from SANS reveals the dangers lurking in the bowels of MS Access, Word 2000 and Internet Explorer 5.00+. SANS is offering a prize for “quick” fixes, but these bugs are eminently “proprietary”. Microsoft, let's face it, has more than their unfair share of expert programmers schooled in the latest software engineering agendas. Indeed, Microsoft Press offers many bibles on how to develop provably “robust” applications. Yet, in spite of linguistic and developmental tools aimed at detecting the “obvious” errors (memory leaks and array overflows), this fine group of MS programmers lets slip the dogs of...er...dogs.

The epistemological, unanswerable challenge, of course, is to first define “bug” and then, “bug-free”. Even Stephen Hawking's “Brief...” open-ended, temporal historiography may not allow us to test every path through all the if-then-elses

of our sweetly crafted code. And what does it mean to “match” a specification expressed, ultimately, in “natural” language? (As Hawking's cosmos time-reverses to the Big Crunch, will we see our GOTOs mapped as COMEFROMs?)

The success of the open-source approach depends on the critical exposure of the *code* to many unbiased programmers motivated only by the true target, whereas an in-house “slave”, with share options at stake, may be loath to rock the boat.

In spite of which, recent SUID security flaws in the Linux kernel indicate (more depressions) that simple code slips can escape multiple perusal. On the bright side, we have (i) prompt, open acknowledgment and fixes; (ii) no end of scapegoats (*boucs emissaries*)—hands up, whosoever was guilty from the 144,000 remnant. Peter Salus will record your confession/contribution.

Due soon from Prentice-Hall, Bob Toxen's *Real World Linux Security* (see Note 1).

GIMPS

Moving to another aspect of widespread open computing, GIMPS is the Great Internet Mersenne Prime Search (see Note 2). Over 8,000 individual users are engaged in finding ever-larger primes. Way back, Euclid proved that there's no end in sight, for if P is the largest prime, then either $P!+1$ is a bigger bugger or it must have a prime factor larger than P . QED! (Where $P!$ means factorial P , not to be confused with the second “!” which expresses amazement and relief!)

They say if you don't see the beauty of this proof (suitably refined), you'll never be a mathematician.

And, a sure test of your *sensayuma* goes thus: the Sieve of Eratosthenes (another dead Greek) offers a slowish algorithm for listing each prime in ascending sequence. My own Kelly's Sieve lists both composites *and* primes with a single for loop.

The point is whether there are primes between P and $P!+1$ —and in June 1999, Nayan Hajratwala of the GIMPS team hit a new (temp) world record with:

$2^{6,972,593} - 1$

I'll spare you the seven miles of digits and commas needed to **printf()** this number.

Next month: how to help find alien intelligence via the SETI collaborative program at <http://www.setiathome.ssl.berkeley.edu/>. [Join the *Linux Journal*

group! —Ed.] Chances are that some little green mathematicians already know the next largest prime.

Notes

Stan Kelly-Bootle (skb@atdial.net) has been computing on and off since his EDSAC I (Cambridge University, UK) days in the 1950s. He has commented on the unchanging DP scene in many columns (“More than the effin' Parthenon”—Meilir Page-Jones) and books, including *The Computer Contradictionary* (MIT Press) and *UNIX Complete* (Sybex). Stan writes monthly at <http://www.sarcheck.com/> and <http://www.unixreview.com/>.

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

[Advanced search](#)

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

Advanced search

Comparison of Backup Products

Charles Curley

Issue #78, October 2000

Backup of data is probably the most neglected aspect of system administration in small businesses and home offices.

PerfectBACKUP+ 6.2

- Manufacturer: Merlin Software Technologies, Inc.
- E-mail: info@merlinsoftech.com
- URL: <http://www.merlinsoftech.com/>
- Price: \$69 US, \$89 US for box and treeware

BRU

- Manufacturer: Enhanced Technologies Software, Inc.
- E-mail: info@estinc.com
- URL: <http://www.estinc.com/>
- Price: \$245 US

Arkeia

- Manufacturer: Knox Software Corp.
- E-mail: sales@arkeia.com
- URL: <http://www.arkeia.com/>
- Price: Free or variable; Quote

Quick Restore

- Manufacturer: Workstation Solutions
- E-mail: info@worksta.com
- URL: <http://www.worksta.com/>
- Price: Quote

- Reviewer: Charles Curley

Backup of data is probably the most neglected aspect of system administration in small businesses and home offices. Yet it can be critical. If your entire business is riding on the contents of your hard drive, and it goes belly-up, you are out of business. Worse, you may be legally liable. For example, you may be required to meet your payroll within a certain time after the end of the pay period.

For these reasons, what you spend on backup, including software and tapes, is a form of insurance. To give one example, I administer a home office with five computers (two Linux, two Windows 95, and one that multiboots between Linux, Windows NT and Windows 95), and two users. The network is standard 10-base T. I have two tape drives (a Conner/Seagate 4GB Travan and an HP DDS 3 DAT drive) and routinely use two of the products I discuss here.

Four backup programs for Linux are reviewed below: PerfectBACKUP+, Arkeia, BRU, and Quick Restore. The first three are available on the Web. Arkeia comes packaged with Red Hat, Mandrake and SuSE distributions of Linux. Quick Restore is available on CD only.

PerfectBACKUP+

This is the only completely “no charge” program in the lot, and coming into such a program, I was not expecting much. I was surprised, for example, to see support for tape changers, something I would not expect to see on a low-end backup program. However, the program qualifies as “nagware” by giving you a “please register” window when you start up the program (see Figure 1).



Figure 1. The Main Window of PerfectBACKUP+

I was very pleased to see that it detected and correctly identified both of my SCSI tape drives. Whether this facility extends to other interfaces I cannot say. The two drives were not associated with the correct device files in /dev, so I hand edited that. A quibble indeed, compared to the setup problems I have had with other tape backup software.

One thing that is very nice about PerfectBACKUP+ is that the device selection menu identifies the drives by their names as provided by the device. That is very useful; it saves having to remember device names.

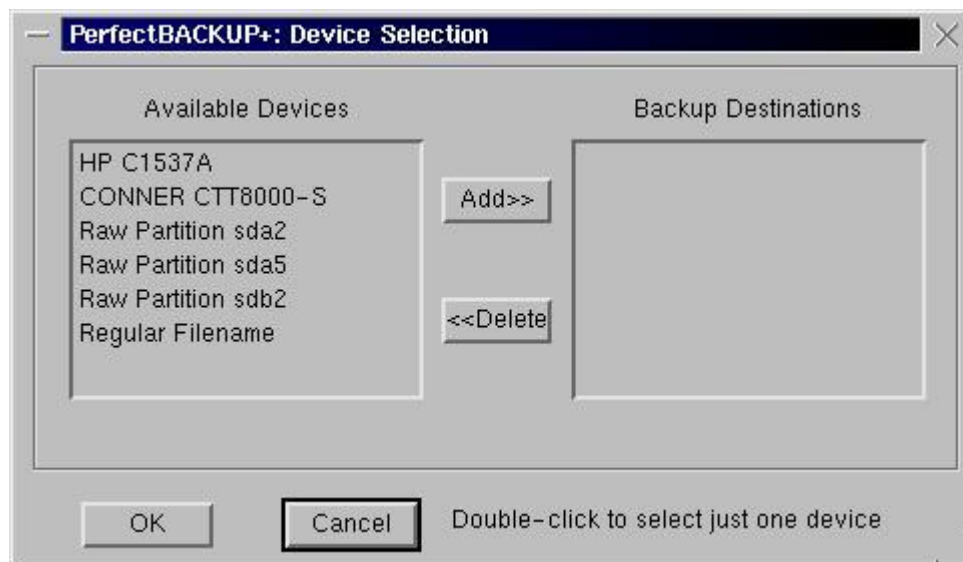


Figure 2. Selecting a Destination Drive for a PerfectBACKUP+

PerfectBACKUP+ provides some easy-to-use minimal acceptance tests, a facility I'd like to see on other backup software. One quibble here is that I'd like to see the software test with larger data sets to better exercise the tape drive. The data set is unfortunately so small as to make the data rate result meaningless.

Defining a backup set (or package, in PB's terminology) is easy: go down the Backup pull-down menu and define each characteristic in turn. Then select the Save As Package menu entry, and give it a name. You can then recall the package and run it all you want. I was able to define a small test backup set and make the backup easily.

Unfortunately, the restoration had problems. I got a cryptic message about "This backup was made from / and you are about to restore it to /test. If your backup was RELATIVE you might not want to do this." Relative to what? Grepping the documentation, I found no explanation of what the term "relative" means. It then asked if I wanted to change the working directory. If I refused, it gave me a cryptic error message. If I accepted, and retyped the working directory I had previously selected, it made the restore where I had asked for it.

This is the kind of glitch that indicates inadequate testing. I can accept that sort of glitch in the GUI, as long as I can work around it. But I am left with the question: what else was not adequately tested? Also, PerfectBACKUP+ left defunct processes on my test computer. This is not encouraging.

Verification is easy. In addition, after the fact—weeks or months—you can pop a tape into a drive and verify it. You can verify the contents of the tape against the original files, or you can run a CRC checksum verification against the tape alone. This is useful for verifying the reliability of your tape drive: remounting a tape is a good test of the repeatability of head positioning, a major casualty of wear in lower-end tape drives like QIC tapes.

Unfortunately, I saw no way to verify a backup as part of the backup sequence. This is something I have been accustomed to seeing in PC backup software since Colorado Memory System's software provided it in the late 1980s.

There are two sets of documentation. One is in HTML, and you read it via Netscape. (What happens if you don't have Netscape?) The documentation screen shots indicate an xterm interface with mouse support; however, the provided interface is a full X GUI. The other documentation is available from pull-down menus on the GUI. It has no search function, and uses a fixed font which is vanishingly small if you have a high-resolution monitor. The two sets of documentation contain different material. Neither one documents the command-line interface, which is a pity.

I must reluctantly conclude that PerfectBACKUP+ is not yet ready for prime time. It needs serious testing in the GUI and a lot of cleanup in the documentation. Because it left defunct processes on my system during my tests, I don't trust it. PerfectBACKUP+ has some excellent features, like identifying drives by name. I hope Merlin fixes its problems and makes a more solid version available.

BRU

BRU is a modern replacement for tar (Tape ARchive). Many of the command line options for BRU are identical to those for tar, which means you can upgrade from tar to BRU fairly painlessly. In addition, XBRU, the GUI, is merely a front end: all XBRU does is build an appropriate command line and hand that off to the command-line executable.



Figure 3. The Main Screen of XBRU

One feature that is missing from BRU is the ability to use the GUI to build a command line, then export it for use in a script. This would be a very useful tool for the command-line-challenged.

The help is all on-line at EST, Inc.'s web site. This is great if you don't mind going on line to read your docs, because they are always the latest. What happens, though, when EST releases a new version? Will your help still point to the web pages for the version you have? Hey, guys: not everyone has a T1 connection to the Net.

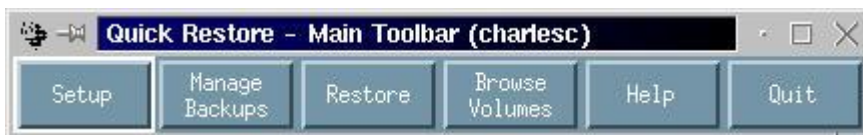


Figure 12. Listing the Contents of a Volume in XBRU

Installation is straightforward. I got the demo version from the EST web site. It comes in a tar file. Copy it into its own directory, untar it and execute `./install`. The installation script asks for information about your tape drives. It will ask you for the capacity (uncompressed) and rewind and non-rewinding device names. The devices ran with no further changes. However, there are sample device definition files for a number of drives on EST's web site.

You can run multiple XBRUs and simultaneously access different tape drives. This is great for backing up to multiple tape drives, but does not give the flexibility of Arkeia's flows.

Verification of a backup is very useful. EST thinks you should do it every time to verify the backup, and that's a good idea. Their autoscan feature makes it easy to build into scripts.

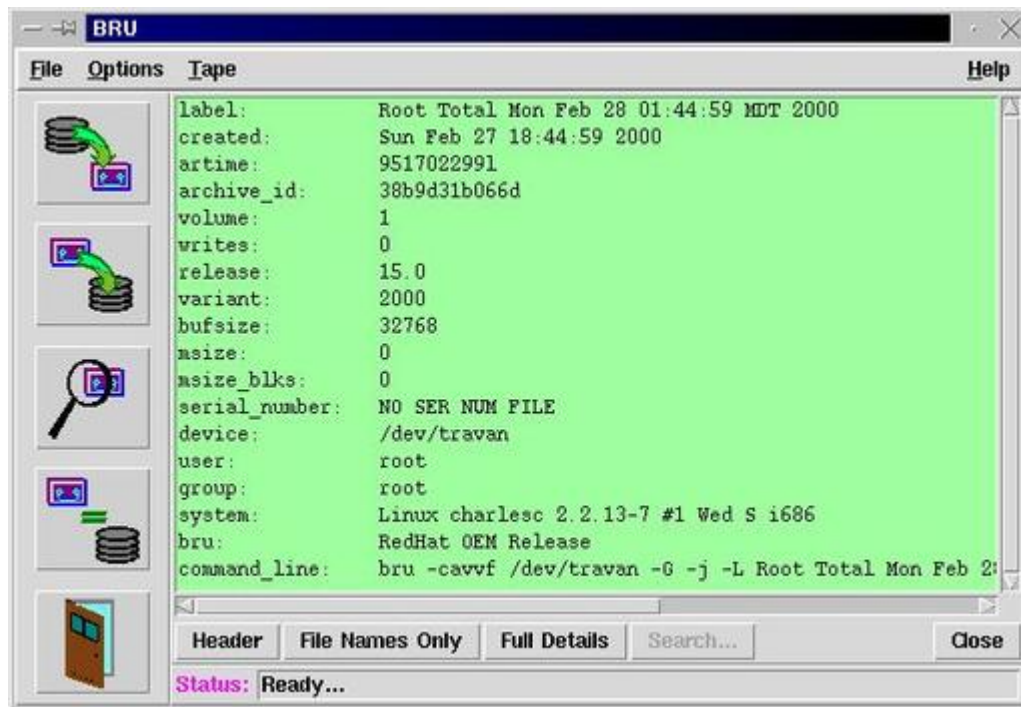


Figure 13. A Tape Header in XBRU

The professional version, which I did not review, will provide a database of files backed up, thereby eliminating the occasional need to read multiple tapes to find a given file.

One thing missing from BRU is support for tape changers. As I have not worked with tape changers, I can't comment on their reliability. But I'd rather have high enough capacity tape drives and arrange my backup schedule so I don't need a tape changer. Unfortunately (for system administrators, if not for users), the ability of hard drive manufacturers to wedge capacity into their products seems to exceed the ability of tape drive vendors to add capacity to theirs. So there may be a tape changer in your future.

Arkeia

Next in line is Arkeia, from Knox Software. Arkeia has a three-part design. The server resides on the machine to be backed up, and writes the backup with local file permissions and bits. The server backs up and restores files; the client runs one or more tape drives on a machine; and the user interface provides user control over the other two parts. The user interface comes in two flavors, command line and GUI.



Figure 7. Selecting a Windows 95 Registry for Backup in Arkeia

One implication of this design is that Arkeia's Windows server, which runs as a native Windows process, can back up the registry as metadata, not as just another file. However, there has been some question about restoring registry backups on the Arkeia user's list, so if you run Windows in your shop, make sure you can do a full restore of the registry on a test computer before you are satisfied with Arkeia's performance.

There is nothing that says that any one component has to reside on the same machine as either of the other two. In my setup, I run both the user interface and the tape client on the same machine. I also run servers on that machine and each of the other four computers on the network.

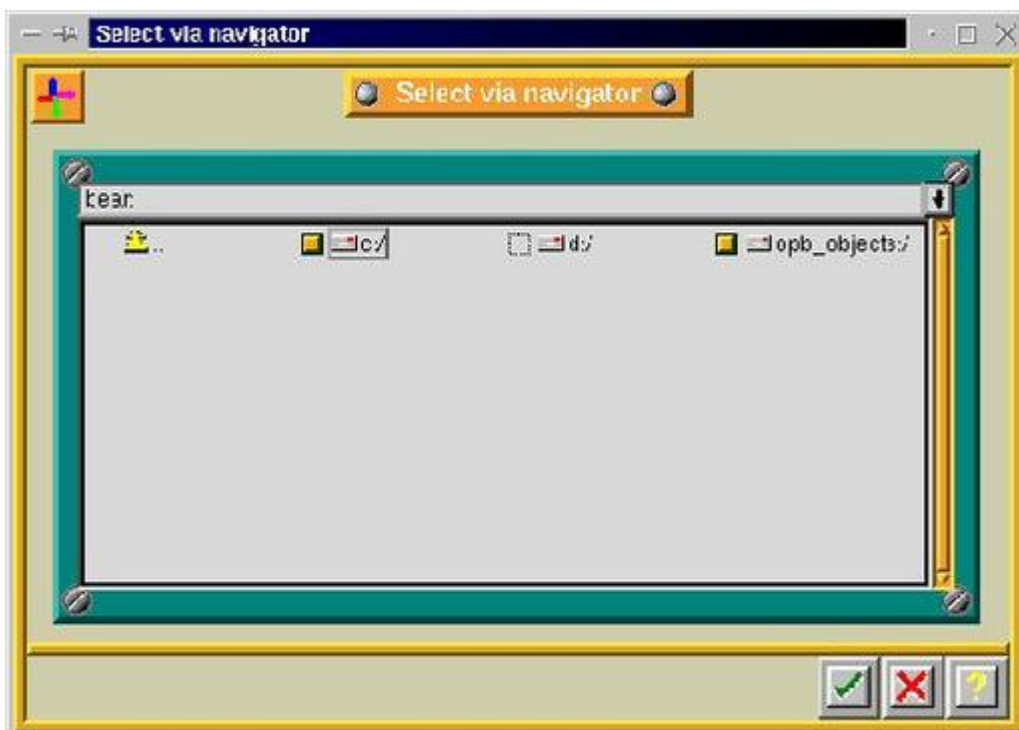


Figure 8. Selection Menu for the Root of a Windows 95 Computer in Arkeia

This architecture makes possible what Knox calls "flows". Most backup software simply walks a list of directories to back up. (The brain-dead backup applet that comes with Windows NT is a classic example.) Arkeia has each server feed its data to the client, so that multiple machines are backed up simultaneously. In my environment, each of the five computers is represented by a "flow". The result is very fast backups: to my HP DDS 3 DAT tape drive, I get backups at a sustained rate of 57MB a minute, or 12GB in under four hours. That data rate keeps the tape drive streaming, which is important for tape and drive longevity. While I haven't tried it, you should be able to back up multiple machines to multiple tape drives. Under those conditions, Arkeia should optimize data flow to keep all of the tape drives streaming.

Arkeia builds a database of files as it saves them to tape. This is great because you can browse the contents of your tape library without having to mount each tape in sequence. It also has a weakness: Arkeia does not back up that database itself. That means if the file system on which the database resides goes south, you have to restore it from tape by reading each tape since the last total backup. My solution to this weakness is to use BRU to back up that file system.

When I first reviewed Arkeia for *Linux Journal* (April 1999, <http://www.linuxjournal.com/issue60/3166.html>), I reported excellent customer support. While I have not used the customer support recently, I must report some grumbling on the list about it. This grumbling indicates that Knox's customer support has gone downhill since I wrote the review. Perhaps the folks at Knox think that a list is an adequate substitute for good customer support; it is not.

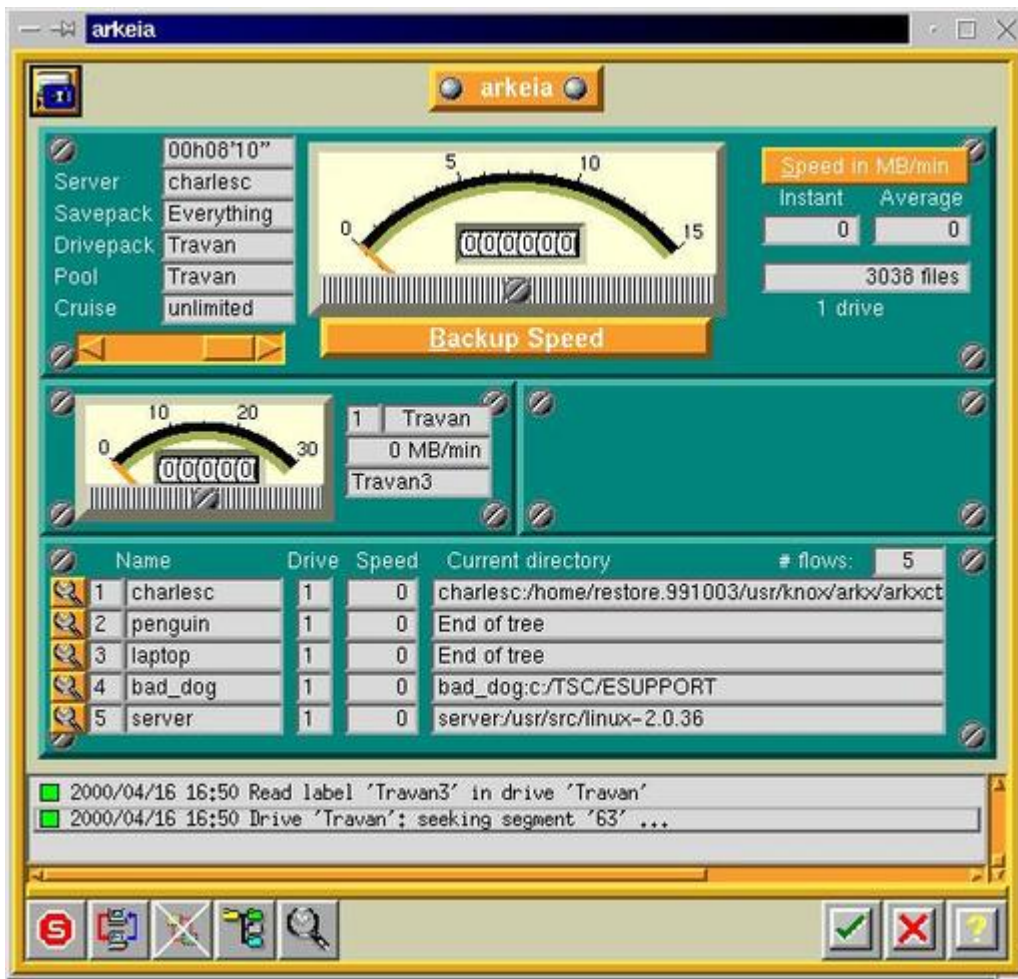


Figure 10. An Arkeia Backup in Progress

Setting up Arkeia initially is a lot of work. You first define everything down to the individual tapes, which are lumped into savepacks, then the drives, etc. Defining a backup is a matter of selecting the savepack and the files to be backed up. Once you have done that, you can run a backup at any time, manually or at pre-defined times. The complexity of initial setup is the price of Arkeia's extreme flexibility. A "wizard-style" interface would be easier to use than the potentially confusing menus Arkeia uses. Short of that, the manual walks you through the process of setting up and running null device backups to verify your installation.

Arkeia is probably the most ported of the lot. If you have more than Linux and Windows NT, look at Knox's web site for a list of supported OSes.

One thing missing from Arkeia is the ability to verify a tape. Neither verification by comparison nor by checksum is available. All I have been able to do is a partial restore to another location and then comparison against the original, a process that verifies only a portion of a tape. Of the four products reviewed here, Arkeia is the only one with no form of verification.

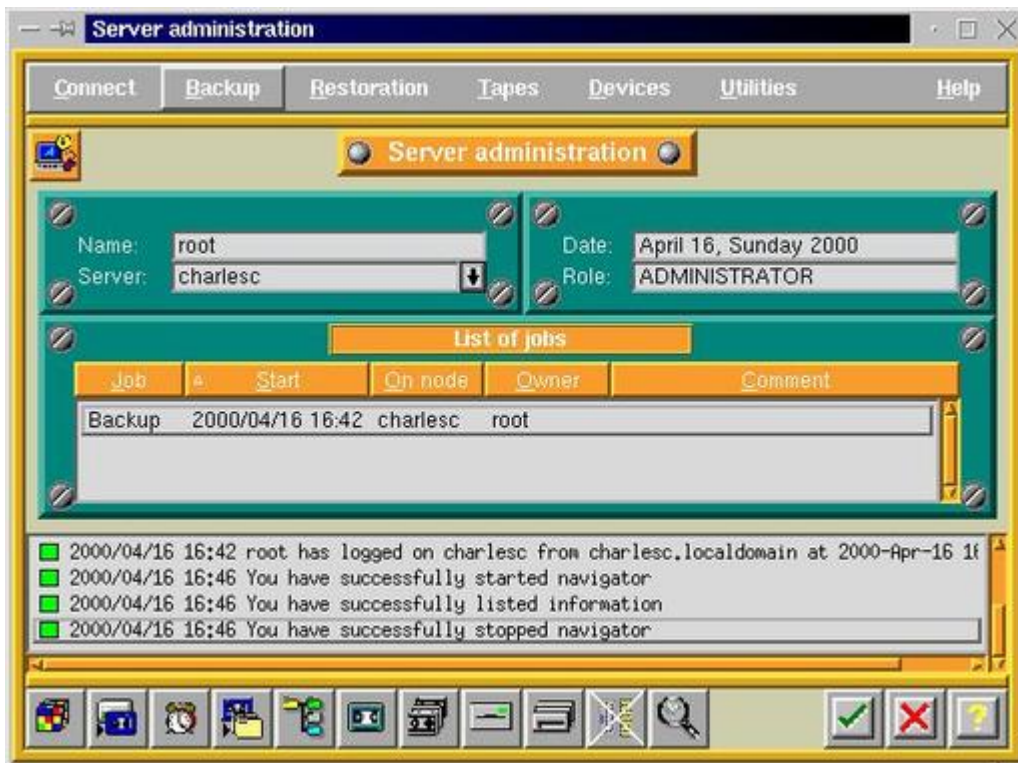


Figure 11. Arkeia's Main Window, Showing a Backup Job in Progress

Arkeia is priced on a per-seat basis. For a quote, visit the Arkeia web site. You can download a "personal" version. The latter is restricted to one tape drive and a fixed number of clients.

Quick Restore

Far and away the high-end product here is Quick Restore, from Workstation Solutions, Inc.

Most Linux users are accustomed to downloading what they want from the Net and trying out the package then and there. Workstation Solution's evaluation process is quite different. Evaluation copies are available on CD only. When you request a CD, the company schedules an appointment to make the installation. At installation time, a technician walks you through the installation. The request process involves some e-mailing back and forth. During my efforts to get an evaluation copy, someone (probably me) dropped the ball, and it took me a while to get the CD and make an appointment. Intentional or not, this process has the effect of filtering out casual inquiries and casual users.

The actual installation process is straightforward. With most installation software, there is either no help, or the help doesn't answer the questions you want to ask. With a human being on the other end of the phone, you can ask any question at any point in the process, and get an answer. So this process represents a much more user-friendly process.

The main window (see Figure 11) is a simple six-button toolbar. Each button triggers a suitable separate window, in classic X programming style. One result is that you can quickly end up with a plethora of windows on your desktop. Another is that you can move from window to window easily, unlike the typical Windows application that only lets you use one window at a time.

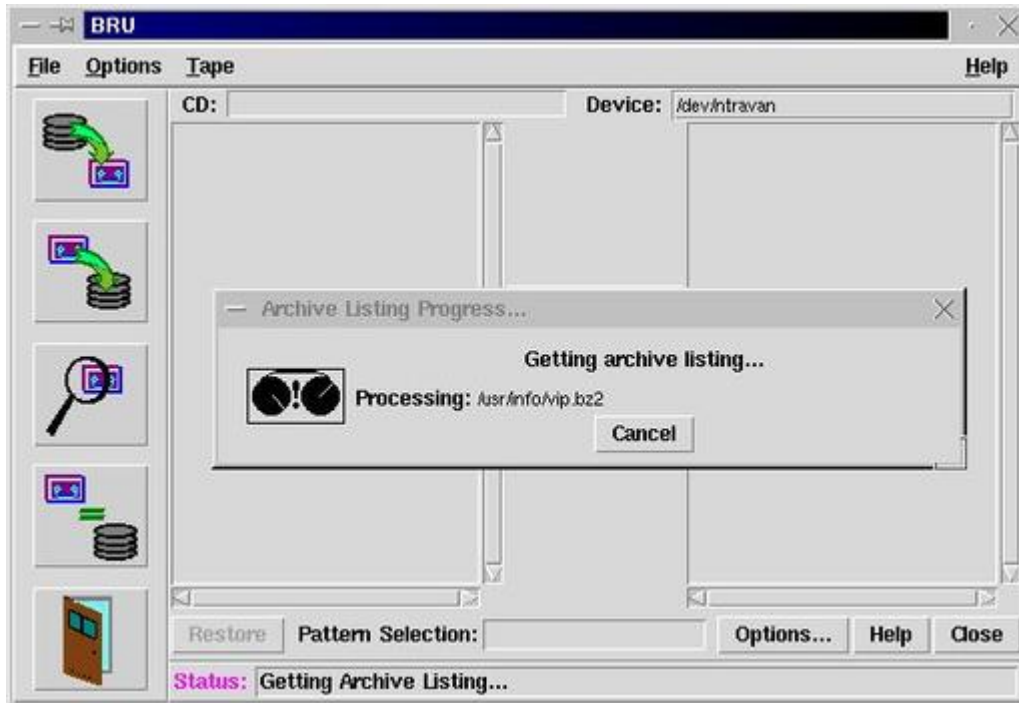


Figure 12. Selecting Files for Restoration

Restoration is GUI-driven (see Figure 12). As with Arkeia, the database of backed-up files is organized in a tree that reflects the same structure as the backed-up file systems. Click on one or more files to select them, then select the volume from which to restore, insert the appropriate tape (unless you have a tape changer) and away you go.

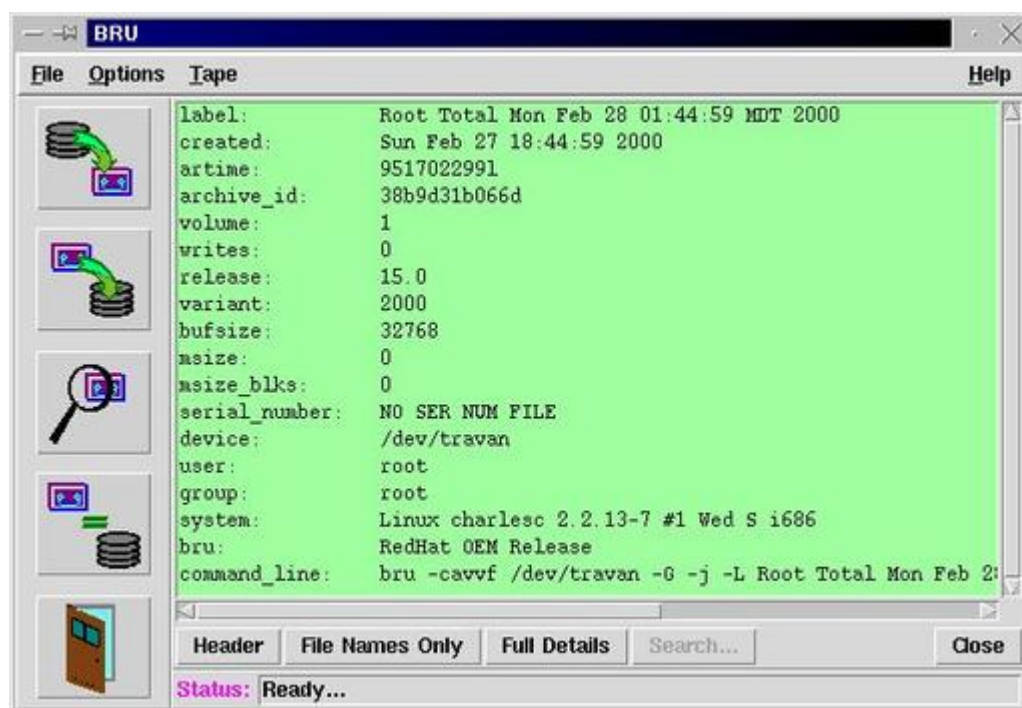


Figure 13. Editing a Dataset in Quick Restore

Unlike Arkeia, backups are defined with a scripting language (see Figure 13). Quick Restore has a built-in editor for the scripts. The bad news is that the editor is a brain-dead CUA interface editor, rather reminiscent of MS-DOS' EDIT.COM. The good news is that the files are plain vanilla ASCII text files you can edit with any editor. Alas, if there is a mechanism for selecting an external editor such as vi or emacs, I missed it in the documentation. The scripting language is clear and well documented. I was able to create data sets in quick order using emacs. The built-in editor has a validator, which you should use often as you edit.

Once you have defined a data set, it is necessary to schedule a backup with it. You can schedule backups daily, weekly, monthly or annually. There is no interactive backup facility; instead, you schedule one for a minute from now and wait. At the time I installed Quick Restore, the technician walked me through this process (except for editing with an external editor) with a small data set and a test backup.

Quick Restore also has a steep learning curve, but I think not as steep as Arkeia's. Again, that's the price you pay for flexibility. You can start using Quick Restore immediately by carefully substituting qtar for tar in your existing scripts.

Quick Restore also provides a number of command-line tools. One is qtar, which will read and write tar format, and supports its command-line options. These can be combined into shell scripts and sample scripts are provided. Indeed, the GUI is a front end for the command-line tools. It would be nice if

the GUI could export command lines for use in scripts. Another tool will tell you a lot about your tape drive, including (if the drive supports it) whether it is time to clean the drive.

Quick Restore is clearly intended for large shops, server farms, data farms and similar high-data installations. It does not even support the Travan tape drive. It does support my DDS 3 tape drive, but the DDS 3 is in the low end of the tape drives it supports. The difference, apparently, is that the DAT drive is capable of a true random seek into the tape, where the Travan must be read sequentially to get to a target block.

I found both the on-line documentation and the printed documentation to be excellent. They are not duplicates, and each serves their place well. You should read the printed documentation before you install the software. I did not, and we got through the installation fine, but there is information in the manuals that might have helped. Indeed, the printed manual is a good introduction to backups in general.

Quick Restore, like Arkeia, runs a background process on each machine it backs up. At the moment, Quick Restore supports Linux, a number of Unices and Windows NT. It will back up Windows 95/98 boxes but only over SMB mounts. This has the disadvantage of backing up the registry only as a standard file, which means the backup administrator has to remember to add it to backups. You should always back up the registry, as this is the only work-around for an incredibly bad operating system design bungle in Windows.

Of the four vendors whose products are reviewed here, only Workstation Solutions uses the Network Data Management Protocol (NDMP) to communicate between clients and servers. This emerging Internet standard should eventually (as more vendors sign up) allow interoperability between vendor products, and that should help greatly in heterogeneous environments.

Quick Restore is not cheap. Workstation Solutions quoted me \$7,250.00 for a backup server, with an additional \$1,595.00 (US) for an annual service contract. Linux and FreeBSD clients are no charge. NT clients cost \$2,250.00, with an annual service contract at \$495.00. There is no "home use" option like that for Arkeia. For that kind of money, customer support had better be excellent! Other than the installation walk-through, I have had no contact with Workstation Solutions' customer support, so I cannot tell you how good it really is.

Tape Formats

PerfectBACKUP+ uses **cpio** format, while Quick Restore uses tar. Both were developed way back when, and may not provide the data integrity of more

modern formats. BRU, on the other hand, clearly uses its own format. The advantage, according to EST, is better data integrity; the disadvantage is that you must use that tool to restore your data.

As far as I can determine, Arkeia has its own tape format. They do provide a command-line utility that allows you to recover a directory of a tape. You can then use this directory to restore from that tape.

Conclusions

The marketplace is wonderful. We see here solutions to the problem of backups from vendors in all price ranges and all levels of support; no Post-Awful "One Size Fits All" mentality.

One concern I have for backup programs is Windows NT backups. Windows 2000 has more file descriptor bits than Windows NT. For example, a file may be a sparse file. Also, Windows 2000 has file system data structures called reparse points. Are the backup programs that handle NT Windows 2000-aware? I don't see anything in the literature of either of the two NT-aware programs reviewed here that indicates one way or another. I do not have Windows 2000, so I was unable to test this point. Workstations Solutions told me they are working on it. Absent any claims that Arkeia support Windows 2000, presumably Knox also is working on it.

If you are spending other peoples' money, or have the budget to buy software but don't have the staff to write your own configurations, and you don't have any Windows 95 or 98 machines to back up, go with Workstation Solutions' Quick Restore. Quick Restore is far and away the best documented of the four products reviewed here.

If you do have Windows 95 or 98 machines to back up over a network, and are willing to learn and use a slightly confusing graphical user interface, get Arkeia. Arkeia is also excellent for sites with a lot of data to back up. The steep learning curve gives you the flexibility to define a lot of different backup configurations. This in turn allows you to get the most out of your tape drives by keeping them as busy as possible.

In one area, Quick Restore is superior to Arkeia. If you have multiple databases to back up, you want to shut down each one for the shortest period possible. Short of exporting the entire database, and then backing that up, the ideal is to shut down each database, back it up, and then fire it up again, in sequence. Arkeia allows you to execute a script before the backup, and another after it. To minimize database downtime, you have to define a separate backup for each database, a nuisance at best. Quick Restore lets you insert scripts at any point in the backup process, which means you can back up an entire computer, and

only have each of its databases down long enough to back it up. BRU's GUI has no facility to run scripts at all, but the command-line interface is excellent for writing your own. PerfectBACKUP+ does not appear to have any facility to run scripts.

EST, Inc.'s BRU Professional appears designed to compete in the high end, along with Arkeia and Quick Restore. Unfortunately, it was not publicly available in time for this review.

The BRU-PE personal edition is an excellent choice for backing up your own Linux machine and other machines over NFS or Samba.

For a small- or medium-sized shop, I would seriously recommend using Arkeia to back up the network, and then back up the tape server with BRU. For complete protection, I would add EST's Crash Recovery Utility or their QuickStart Data Rescue. As I have an HP One-Button Disaster Recovery (OBDR) tape drive, the CRU looks very attractive to me.

For home networks involving Windows, I would look at Arkeia's shareware version, supported by backing up Arkeia's database with BRU-PE or tar.

Resources

Table 1. Features Summary

email: ccurley@trib.com

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

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

[Advanced search](#)

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

Easysoft Data Access Middleware

Jon Valesh

Issue #78, October 2000

Easysoft provides interfaces between data and the applications that use it.

- Manufacturer: Easysoft
- E-Mail: sales@easysoft.com
- URL: <http://www.easysoft.com/>
- Price: Free for single users, commercial licenses vary
- Reviewer: Jon Valesh

Easysoft provides interfaces between data and the applications that use it. With Easysoft's products, you can access any ODBC data source from just about any computer system, and you can allow access to many Linux databases from ODBC clients. To use real-world examples, you can access Access on a Win98 desktop from your Linux web server, or your MySQL database from Excel.

Very few assumptions are made about what you will do with that data once you have it. Use it with PHP or iHTML, write custom C code, access it from Python or Perl or a Java applet; Easysoft doesn't care.

Easysoft provides a combination of their own ODBC-ODBC bridge software to allow access to remote database servers, and unixODBC 1.8.10 to access Linux database software. Access to SQL databases is through Easysoft's SQL-ODBC gateway package. This standardization on ODBC for all database access gives two important benefits: your programs will be totally free of database-specific dependencies, and you can seamlessly access multiple databases without special code. Not only databases, you can access any data source that has an ODBC driver.

Installation

On Easysoft's web page (<http://www.easysoft.com/>), they claim you can install their ODBC-ODBC bridge in one to two minutes. They aren't far off. It actually took closer to five minutes, but perhaps I was having a slow day.

Easysoft supports Linux as just one of about a dozen platforms, and they obviously know having a distribution that works regardless of one's system type is very important. In the Linux arena, they have libc5 and glibc 2.1 versions for Intel x86 and Alpha processors. Installation is done without the use of a system package tool and requires superuser access, because several new libraries are added and files in your /etc directory are changed. All files are placed in the /usr/local/easysoft directory. If downloaded from Easysoft's web page, the software is packaged in a compressed tar file and installation is a two-step process. First, you must unpack the obscenely long-named tar file, change to the newly created directory and run the **install** script.

The install script is a model of how self-installing Linux programs should behave. You are asked a series of questions about which components you would like installed, what directory to place the files in and how everything should be set up. Given the quality of the installation script, very little would be gained by using distribution-specific package tools, and Easysoft's already daunting array of download options would increase unnecessarily.

One stage of the setup includes running Easysoft's license management software. This application collects personal information about you and how you will use the product, and contacts Easysoft's servers through the Internet to request a license key. If you don't have an Internet connection, you can print out the collected data and mail it to Easysoft via real-space mail.

The Easysoft's installation script installs unixODBC. Easysoft listens on port 8888 for ODBC requests and 8890 for HTTP requests.

Easysoft's package is made up of separate client/server applications. The clients and servers are a rather mix-and-match affair. You can run a Windows server and Linux clients, or Linux and OpenVMS servers and Java clients (using the ODBC-JDBC bridge), or whatever your application demands. For most users, the servers would be installed on the company database server computers, and the clients would end up wherever the data was needed. This could be your web server, a desktop computer running a database client application, or both. This gives you great flexibility in choosing your data sources, and even allows some unlikely combinations like using spreadsheets on desktop computers as data sources for dynamically updating web pages.

Using the Software

Easysoft's tools are designed to facilitate getting data for your applications, not writing them. In fact, they don't really even provide the API. If you know how to use database tools, Easysoft's products will give you access to a wider range of data sources. If you don't, they provide some tutorial information to point you in the right direction, but that's about it.

The APIs you use are specific to your language or development environment, not Easysoft's product. Most aren't that complicated, but it is totally a matter of what tools you are using. If writing PHP scripts, you connect to your databases with the PHP ODBC support. A command such as `@odbc_pconnect(<datasource>, <username>, <password>)` establishes the connection to the data source, returning a string if the connection is successful. Once connected, you request data by assembling a string with your query, and running another API command to execute it. If you are writing in Perl, you use the DBM::ODBC API to access the data sources. If your development tools include an ODBC API, chances are you can use Easysoft's tools to provide access to a wider range of data sources.

Licensing

The Easysoft OOB client is free for use with any OOB server you have a license for. That means you can install clients on any computer in your office without having to worry about specific licenses. Oddly, you are licensed to use the OOB client only with servers you own and control. You can't use the OOB client to access another company's server. Easysoft's own demonstration application seems to cause people to unwittingly violate the client license. There is also a limitation on incorporating the client in any program or system you receive payment for the use of without written permission. I'm sure that wasn't the intent, but it could possibly be read to say that you need written permission to charge for access to a web page with scripts that use Easysoft's client. This would limit its usefulness in some applications.

Documentation and Getting Help

Easysoft provides very complete documentation to help you get started doing real work. Included with the software are instructions for getting Apache and PHP installed, or working with Applixware, or Perl DBD::ODBC, or ColdFusion, or a few other commonly used configurations. The FAQ contains a lot of questions, some of which I can't imagine are all that frequently asked. There are also tutorials and sample applications to get you started writing code.

Easysoft maintains a newsgroup for support questions, with technical staff monitoring the group and answering questions.

Conclusion

Easysoft offers a lot to developers who need to add uniform data access to existing programs, or develop data-aware applications for web or client/server deployment. The wide range of both clients and servers allows you to integrate complex networks without writing a lot of special code. Easysoft allows you to download and try out their software before you buy. With their thirty-day trial, you can access all of the features, including multiple database servers. A single-user, single-database license is free for personal use.

Born at the beginning of the microcomputer age, Jon Valesh (jon@valesh.com) has pushed and has been pushed by computers his entire life. Having run the gamut from games programmer to ISP system/network administrator, he now occupies himself by providing technical assistance to ISPs and small businesses whenever his day job doesn't get in the way.

[The Good/The Bad](#)

email: jon@valesh.com

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

[Advanced search](#)

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

[Advanced search](#)

Magic Enterprises Edition 8.3 for Linux

Jon Valesh

Issue #78, October 2000

Magic Software Enterprises, or Magic, for short, has been developing multi-platform data access middleware and on-line application development tools for UNIX, AS/400, OpenVMS and Windows users for years.

- Manufacturer: Magic Software Enterprises LTD.
- E-Mail: sales@magic-sw.com
- URL: <http://www.magic-sw.com/>
- Price: Development Kit, \$299 US for single user
- Reviewer: Jon Valesh

So, your company has a web page and a corporate inventory/product/sales database, and you are the one who makes it go. You're the wizard, the head web-technology guru, the person everyone turns to when something has to work. And your boss just came into your office asking why on-line orders from your web site are being e-mailed to salespeople for manual processing rather than interfacing directly to your existing sales database. And, why you've got customers ordering products that your inventory system says you don't have; why customers can't search for products on your web page; why you've got customers calling customer support to find order status information that is easily available from, you guessed it, the database. And on top of all that, your manager wonders why a web designer is manually building web pages for each new product that goes on the web page, even though (you guessed it) the database has all the information the web page needs.

Or maybe you are the boss, and you are doing the asking. Either way, money is being spent to duplicate information from your database on your web page, and jobs you've got software to handle are being done manually. The result is subpar service and a subpar web page, at extra cost.

Worse, potential customers see your web page every day. If it isn't the slickest page around, and if your prices aren't the lowest, those customers will go elsewhere to spend their money. Slick and cheap wins in the e-commerce world. If you can't get your costs down and your service up, *now*, your customer base will go elsewhere. You don't have time to wait for a C programmer to develop a custom database module for your web page—you need a solution that works today.

Or maybe you are really starting from zero, and you don't have a database system at all. You don't know what your inventory is until you look, or what an order's status is until you find the paperwork on someone's desk.

That is where data access middleware and on-line application development systems come in. Simply put, data access middleware provides a bridge between your user-level applications and your database. On-line application development systems allow you to rapidly develop web or client/server front-ends for your existing database, or custom databases to match your front-end needs.

Magic Software Enterprises, or Magic, for short, has been developing multi-platform data access middleware and on-line application development tools for UNIX, AS/400, OpenVMS and Windows users for years. Now, Magic's software, Magic Enterprise 8.3 and Magic Toolkit 8.3, are available for Linux, recognizing Linux's growing impact in the e-business and e-commerce market. Magic Enterprise 8.3 provides the back end, and Magic Tool Kit 8.3 provides the rapid development environment for e-commerce and other database-intensive applications. Between the two, you'll have everything you need to make your boss happy, at least about your database/web site integration.

Installation

Installing Magic Enterprise v8.3 and Magic Toolkit on my Red Hat 6.1 test system started out easy. There isn't much to it. Magic supports Red Hat 6.0, 6.1, and SuSE 6.2 systems, distributing their software in RPM package format. To install, you run **rpm** with a few options, like the package file name, and most of the work is done for you.

Not all of it, though. After the installation, you must expand the user files into your user directory, and manually start the server daemons that make everything go. The instructions detail the process well, but lack troubleshooting information. If anything goes wrong, you are stuck with either figuring it out yourself, contacting customer service or going to Magic's on-line forum and asking other users.

Though the software is supplied as RPMs, it installs into the /usr/local directory on your system as though it were manually installed. That is a little odd, but causes no problems when using the software.

The folks at Magic are seriously into the idea of making money from their software. There is nothing wrong with that, but preventing unauthorized use on the Internet is a trick. Magic's solution to unlicensed users is a license server installed along with their software to keep people honest, or at least harass them if they aren't. It works; the license server gave me nothing but trouble—a testament to its honesty-detection algorithm, but rather frustrating if you are me.

The trouble continued until I gave up and started to e-mail Magic's technical support people. Then, in a perfect example of computer perversity, I ran the troublesome software one final time to copy the exact error message into my e-mail, and it worked! I have no idea what made it work, and I have no idea if it will work for anyone else, but I was happy. (Being serious, if your system host name has any periods in it, e.g., is a fully qualified domain name, the license server will fail with a totally unhelpful error message.)

The Development Environment

Magic provides an X-based integrated development environment with a lot of handy features such as a concurrent version control system, a WYSIWYG user interface builder and a project management system to make it easy to switch your development focus.

The interface is very database-like in feel; applications, database tables, user-interface elements, *everything* is viewed as a table. This makes sense, considering that Magic's primary focus is working with databases, but it will seem unfamiliar to people used to more conventional programming environments. Of course, Magic isn't really a programming environment. It is an application-generation environment, so a lot of things will seem odd if you are expecting to start coding.

Everything is a table, including the list of applications themselves. When you go to create a new application, you start in the Settings menu, and configure the application you want. Once it is configured, you go to the File menu and open your new application.

The development environment is totally separate from traditional UNIX/Linux development tools. The version control system is specific to Magic, the debugger is a Magic debugger, everything is Magic. This allows Magic developers to seamlessly cross platform boundaries. When you develop Magic

applications, it doesn't matter if you are working on Linux, Windows or an AS/400: the application and the environment are the same.

Writing Applications

When it comes to writing applications, the language paradigm is everything. If your language paradigm doesn't match the problems you face, everything is harder. Magic deals in databases; Magic's tools act like database tools. Magic's paradigm is the database. For solving database problems, Magic works very well, but programmers should be prepared to shift world views while learning the system. It is not like programming in Perl.

Magic provides a web tutorial to guide you through developing your first on-line application. If you don't go through the tutorial, you will be lost from minute one. It isn't that Magic is complicated or hard to use. Magic is not intuitive. It isn't hard, either, but the user interface and application development paradigm are unique enough that you can't count on experience to let you fake knowledge of the system.

In Magic, you create a database, define the queries and then design the user interfaces. Magic's intelligence shows because it encourages you to design your database before you worry about the trivial stuff like how the web page is going to look. It is also handy because by the time you get to the user interface, almost everything is done; and since Magic automatically generates and publishes the HTML, by the time you finish the interface, *everything* is done. Following the tutorial takes about an hour, and when you are done, you will have the start of an e-commerce site that would be an improvement for many on-line retailers. With Magic, a demo and a working program may be just a link apart.

Putting Your Work to Work

Magic's strength is in getting real-world web and client/server applications up and running quickly, and running fast. Calling Magic "Data Access Middleware", or an "Application Development System", or any other simple one-line name risks losing sight of what Magic is for: to make database interfacing easy and scalable. Even that risks oversimplifying things. Perhaps it is: to put the thousand-order-per-minute e-commerce site within reach of the three college students and a venture capitalist that are going to revolutionize on-line sales. Or, to make the ten-thousand-billion-product on-line catalog a one-month project, rather than a life's work. Maybe even to make securely incorporating on-line ordering to your existing database system a seamless extension of what you are already doing, rather than a redesign of your entire business model.

The question you should ask yourself is whether the cost of a super-slick, do-everything database/application generator system will pay for itself the way you plan to use it.

The Magic Enterprise 8.3 server allows you to expand your existing capabilities or add new capabilities, keeping the process as seamless as possible. From interfacing to multiple database systems to load balancing across servers to dynamically partitioning client/server applications, Magic makes the real-world job of keeping a big system running a manageable task. Magic knows this, too, and they know the value of rapid development and scalable deployment. The Magic development kit for Linux is available for an introductory price of \$299 US, and single-user licenses for Magic Enterprise 8.3 are free, but licenses for additional users will cost you. When evaluating application development environments and run-time systems, you've got to look at what you will save and what new abilities you will have. If your project would take six months using conventional and free tools like PHP3, or a month using Magic, the extra licensing expense may be more than offset by the savings in development costs.

And if getting big database access jobs done fast is important to your bottom line, Magic may be just what you need.

The Good, The Bad

Born at the beginning of the microcomputer age, **Jon Valesh** (jon@valesh.com) has pushed and been pushed by computers his entire life. Having run the gamut from games programmer to ISP system/network administrator, he now occupies himself by providing technical assistance to ISPs and small businesses whenever his day job doesn't get in the way.

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

[Advanced search](#)

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

RAD for Linux: A Review of Omnis Studio

Nick Wells

Issue #78, October 2000

Many professional software developers rely on Rapid Application Development (RAD) products to design, prototype and code applications.

- Manufacturer: Omnis Software, Inc.
- E-Mail: us_sales@omnis.net
- URL: www.omnis-software.com/products/studio/index.html
- Price: \$149 US
- Reviewer: Nicholas Wells

The gcc compiler we all rely on when developing programs for Linux is a standard recognized throughout the entire UNIX world. It's fast, it's straightforward, and it compiles standard C (or C++ for the g++ compiler). But anyone who has written a large makefile knows it isn't a complete development environment by itself. With the advent of some serious tools like Code Fusion and Source Navigator from Cygnus, developing large projects with multiple programmers is getting easier all the time.

Rather than use these tools, however, many professional software developers rely on Rapid Application Development (RAD) products to design, prototype and code applications. Many of these applications revolve around database access; many are created by an organization for internal use or by consultants for vertical markets. These might be billing or database access systems, customer support systems, or pieces that integrate existing web services so all employees can access data using a browser. The issues facing these internal developers are different from those that face someone creating a "boxed product" or a standard open-source project for eventual public consumption. Three examples of these issues include very tight deadlines, a need to integrate with existing IT infrastructure, and a need to run on multiple platforms running in an organization.

The process for completing an internal project might look something like this:

1. Plan the project, including creating a schedule for development.
2. Create a demo (if things are moving slowly) or a prototype (if they're moving rapidly).
3. Develop the actual program.
4. Test it within the department where it will be used.
5. Deploy the program, and begin the revision phases as needed.

Java is the proposed solution to many of these internal development projects. It's a cross-platform language, it works with the Web, and everyone seems to accept it as a good idea. But Java is not always the answer, as developers on many abandoned projects can attest. Microsoft Visual Basic is also widely used for these internal projects.

The problem with so many of these internal projects is that after a lot of work has been put into the planning step, the prototype and development take so long and cost so much that the whole project may become unworkable (or at a minimum, go way over budget). Part of the problem lies in converting a plan into a prototype, or a prototype into a working program. Other problems often revolve around a lack of appropriate tools or talent, or the need to create the program so that it runs on multiple platforms (or the Web—which brings a separate set of challenges). RAD tools are the solution many developers look to. These developer tools combine a graphical interface, database access and additional tools to help code, debug and deploy an end-user-oriented application.

Omnis Studio

While a few high-end tools have been available on Linux for a while—some for \$10,000 or more per developer—those needing RAD tools normally didn't consider Linux. (One promising tool, Borland's Kylix, is still in beta testing.)

A recent entrant in the field of Linux software development is Omnis Studio from Omnis Software (<http://www.omnis-software.com/>). Though this company and product have been around for over twenty years, they began shipping a Linux product just last year. Fortunately, the maturity of their products and the depth of the feature set have carried over very well to the Linux port. Omnis Studio takes a Java-like approach to software development that creates a single cross-platform program in a very powerful graphical development environment.

Omnis uses a run-time engine to both develop and execute programs. If “editing” is enabled in your license, you can create new programs; otherwise,

you can run only existing programs. Amazingly, this engine (without the libraries your program may require) is only about 1MB. Both the small size and the execution speed of Omnis Studio applications are surprisingly good, given that this is an interpreted environment. Engines that let you create and execute Omnis Studio programs are available on Linux, Windows and Macintosh. To show the flexibility of the technology behind this product, consider that you can develop a program on Linux, make changes to the same program file on a Windows system (running a developer license of Studio), then run the newly altered program on a Macintosh. You can think of the “program” as a script that can be run or edited on any of the platforms. Figure 1 shows the architecture of the product.

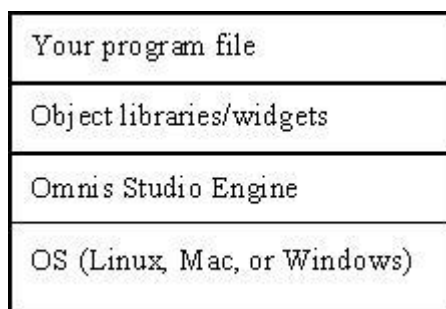


Figure 1. Architecture of Omnis Studio

Longtime users of both Omnis and other tools (Java or Visual Basic) have told me that Studio requires about 20% of the development time of Visual Basic because of the automation Studio includes. Some of these automation features are things like auto-typed variables which are tokenized throughout the project, and single-file projects that collect all relevant information in one place.

Web Wizards

Omnis Software considers Visual Basic to be the main competitor to Studio. The good news is that with Studio, internal development projects which would have required Windows workstations or an expensive UNIX RAD solution can now be developed on Linux and executed on any system with a browser. The interface to Studio is shown in Figures 2 and 3. It's a challenge to capture many features in just a screenshot or two, but you can see some of the widget palettes, properties, notebooks and method editors. Standard tools like buttons, lists, menus and text are included, all tied together for quick results. A scripting language called “notation” is used to add code to objects, but in my experience, little coding is needed compared to something like Visual Basic. The entire product is object-oriented, with object inheritance and powerful class libraries.

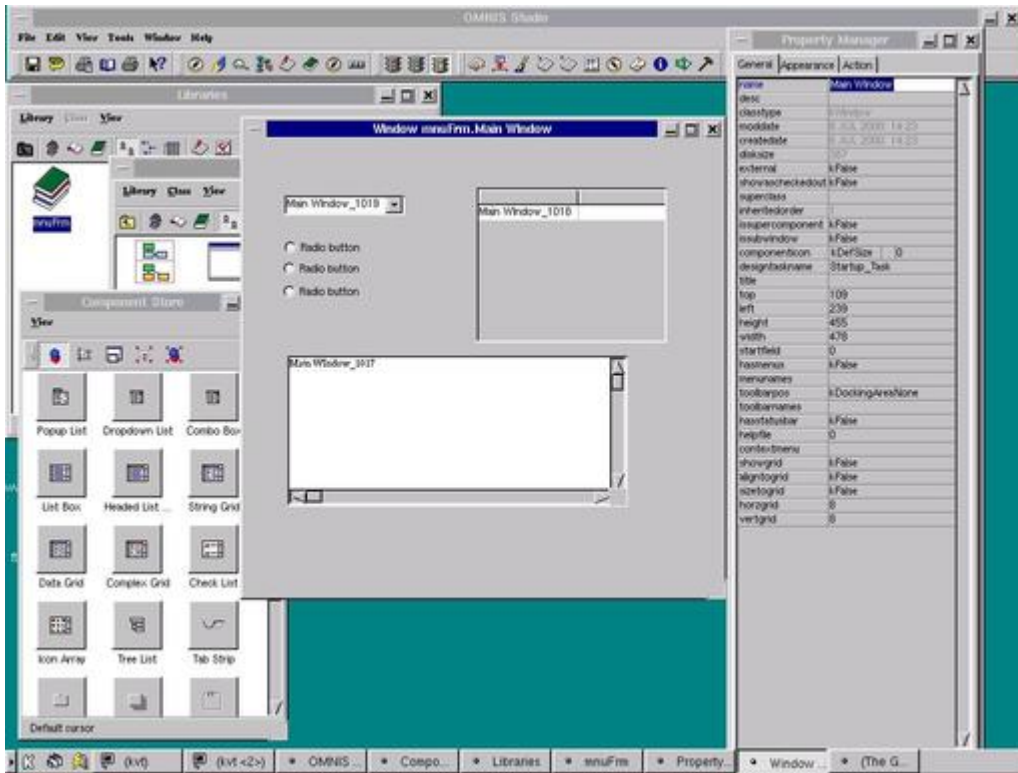


Figure 2. Omnis Studio Interface

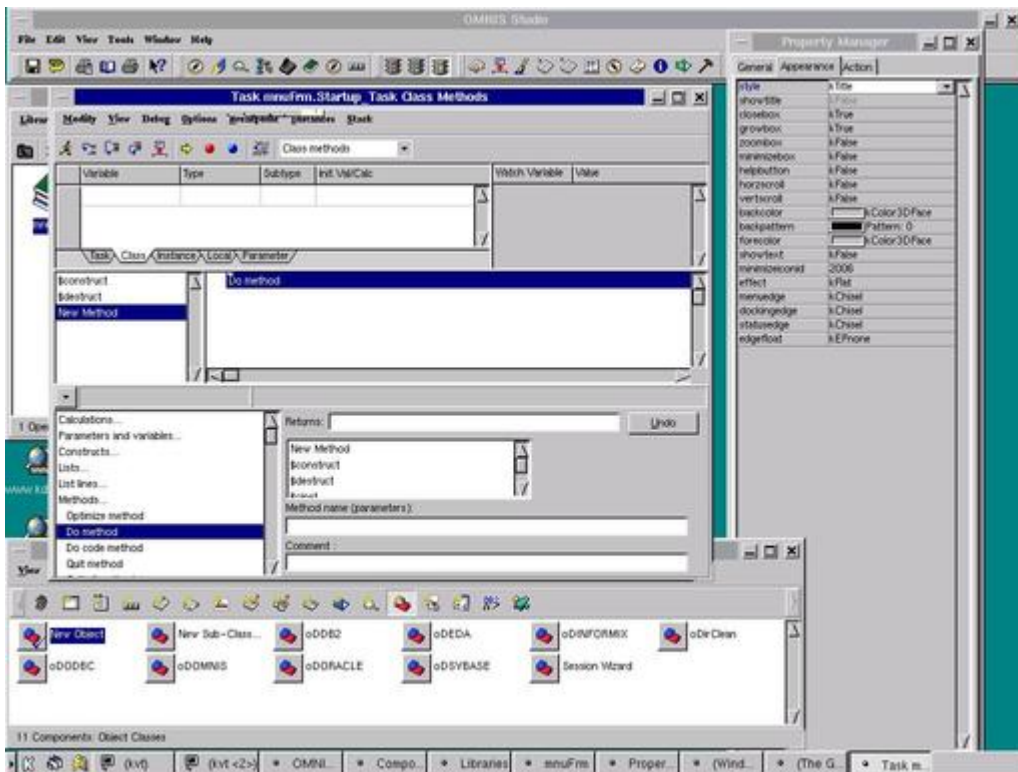


Figure 3. Omnis Studio Interface

Two of the most time saving widgets included with Studio are those for database access and web-based applications.

The Database widget creates a connection to a database within your application, allowing you to view, add and delete records. The database can

reside on several types of servers including Oracle, Sybase, Microsoft SQL, IBM's DB2, or any ODBC-compliant database server. All the details are taken care of for you; just define a form for the fields to appear in, and code the actions to take on each piece of data that an end user works with.

Because the development platform is also the run-time engine, you can use the database widget in real time to manage databases as you develop an application. You can copy schemas (record layouts) between tables or into your program, or move data between tables located on different database servers (even if the servers are running different database products).

Omnis Studio also includes a plug-in you can add to your browser so that any application you write can be used within a browser over the Web. To "webify" any application you've created, just add a few widgets to the application. This does require you to include the Studio application on the system running the web server (Apache on Linux or IIS on Windows), plus you must add the Studio plug-in to each client's browser, but the ability to rapidly turn Linux into a true application server may be reason enough for some developers to give Studio a try.

Prototype? What Prototype?

The biggest advantage I've seen with using Studio is that the prototype I create using the application wizard, standard widgets and a little bit of scripting is a working program which I can begin testing. If my "prototype" requires some tweaking to fit the project description or client requests, I can do that immediately and even re-deploy the application via the Web to people who are already using it. Automatic updates is a feature not all developers will need, but creating a prototype that with a little polishing turns into a working program is something any overworked developer can appreciate.

You can get a full copy of Omnis Studio for \$149. That doesn't include any printed manuals, but all of the documentation (several thousand pages) is included on the CD. Separate printed documentation sets are available if you need one. Several free downloads are available from the Omnis Software web site, including additional database support modules, an evaluation version of Omnis Studio, and the web plug-in (so you can try existing applications at the web site using your browser). A limited edition of Omnis Studio 2.4 is also included with Caldera's OpenLinux 2.4 eDesktop product.

The Catches

Okay, so I'm very impressed with Omnis Studio. But a couple of flags need to be raised as well. First, the learning curve with this product can be steep. Maybe this is best expressed as "easy to understand, difficult to master", which of

course is much like Linux itself. Studio uses a proprietary scripting language and a number of "wizards" and widgets of its own design. All this is similar to products like Visual Basic, but Studio has its own way of doing things. If you're willing to put some work into learning the product, you can generate amazing results; if you just tinker, you're unlikely to even scratch the surface of what the product can do. Omnis is working hard to make it easier to learn their product, and they have a terrific group of experienced developers who help each other via the Net. In addition, the documentation is solidly written and full of complete examples, included on the product CD-ROM.

The other big catch is that this is true commercial software. It comes with a heritage of big UNIX and Windows installations at Fortune 500 companies. Although Omnis has lowered their prices significantly as they enter the Linux market, the product is really intended for internal or consultant-oriented development at medium to large companies. To that end, you must pay a runtime engine license fee for each workstation on which you deploy the application you write. It's a small fee, on the order of \$10, but it's certainly not open source. On the other hand, when you're under the gun to create a powerful program with a ridiculous deadline, it's a price you may be happy to pay.

The Good/The Bad

email: nick.wells@lineo.com

Nicholas Wells (nick.wells@lineo.com) is the director of technical services at embedded Linux vendor Lineo, Inc. He has published about ten books on Linux, KDE and the Web.

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

[Advanced search](#)

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

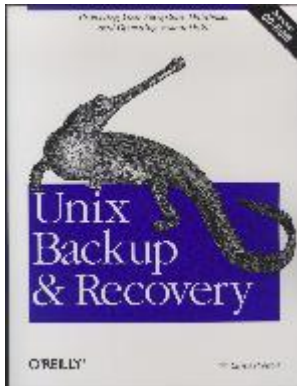
Advanced search

UNIX Backup and Recovery

Charles Curley

Issue #78, October 2000

Buy this book. Now.



- Author: W. Curtis Preston
- Publisher: O'Reilly & Associates
- URL: www.ora.com
- Price: \$36.95 US
- ISBN: 1-56592-642-0
- Reviewer: Charles Curley

Buy this book. Now. Do not pass "Go", do not let your hard drive crash. As soon as you have the book in hand (espresso optional), read Section I, which consists of Chapter 1, "Preparing for the Worst", and Chapter 2, "Backing It All Up". Break out the CD-ROM and see what's on it. Skim Section II, "Freely Available File System Backup & Recovery Utilities". Read Chapter 8, "Bare-Metal Backup & Recovery Methods for Linux". Then, do what the man says. Implement at least a minimal backup system, and build the tools you need to perform a bare metal recovery. Boot the bare metal recovery diskette, and make sure you can read the ZIP disk or whatever you used in its stead.

That was the short review. Any questions?

The Long Review

Most of what I know about backup and recovery I know from having done it. A stint at Colorado Memory Systems writing backup and recovery software for MS-DOS, Windows and various flavors of UNIX didn't hurt. Since then, I've been responsible for backup and recovery at a number of shops. Not least among those shops is my own home network which consists of five computers, two of which are Linux boxes. Since my lady and I earn our livings with our computers, my ability to restore data may represent our ability to earn our livings.

The key lessons I have learned when it comes to backups are: (1) Murphy was an optimist, and (2) when you find out that Murphy was correct, it's usually too late to do anything about it.

Apparently, I'm not the only person who has learned these two lessons. Some people learn them the hard way, and Curtis Preston provides us with plenty of anecdotes about how he and some of his colleagues learned them the hard way. Now, you can read these horror stories and learn from someone else's mistakes. Believe me, that's a much smarter way to learn.

Curtis Preston knows the subject matter at hand with expertise that comes from years of experience in shops large and small. According to the biography on O'Reilly's web site:

The first environment that Curtis was responsible for went from seven small servers to 250 large servers in just over two years, running Oracle, Informix, and Sybase databases and five versions of UNIX. He started managing this environment with homegrown utilities and eventually installed the first of many commercial backup utilities. His passion for backup and recovery began with managing the data growth of this 24 x 7, mission-critical environment.

This book also draws on a network of about 400 experienced consultants called the "Collective Intellect" to fill in gaps in Preston's own knowledge. The result is an excellent book that seamlessly covers several major UNIX versions, including Linux.

The writing style is informal, nonacademic and results-oriented. For example, if you want to use the find command to specify files to tar for backup, you can do this via a named pipe. Preston not only tells you this but shows you the commands necessary to make the pipe and use it. He then gives you the three variants in syntax for the various UNIX systems the book covers. It is detail like this that makes the book an excellent reference work as well as a textbook on the subject.

Some of the subheadings indicate the informal style: "My Dad Was Right"; "Test, Test, Test"; "Don't Skip This Chapter!"; "The Muck Stops Here: Databases in Plain English"; "Trust Me about the Backups"; and (unfortunately inevitable in a book on backup and recovery) "An Ounce of Prevention...".

The book is divided into six sections, each containing one or more of the 19 chapters. The Introduction contains chapters 1 and 2 and deals with general thinking about backups, such as how to organize your backup and recovery plans, what to back up and what not to back up. The introductory chapters emphasize planning and documenting, and rightly so.

The second section concentrates on backup and recovery tools you may already have, like tar and cpio; and tools you can readily get, like amanda. amanda may be worth the price of admission all by itself. amanda is designed for backing up and restoring multiple hosts over a TCP/IP network and has provisions for defining data sets and scheduling backups. It is in the same league as Arkeia and Quick Restore, but comes with source and is free of charge.

Section III delves into commercial backup utilities. As there are many excellent backup tools available for Linux, this is worth a read. Preston does not recommend any specific programs but does give an excellent overview of features to look for and "features" to avoid. The next chapter deals with High Availability. It starts with a definition of the term and then explains why you should be backing it up.

Then, we get into the really fun stuff: bare metal backup and recovery. You've just had a fire, your computer now looks like something Salvador Dalí would paint and runs about as well as it looks. Now what?

There are chapters on SunOS/Solaris, Compaq Tru64 UNIX, HP-UX, IRIX, AIX, and, of course, Linux. The Linux chapter uses the tomsrtbt mini-distribution of Linux and an Omega parallel port ZIP drive to recover an Intel architecture system. Preston also gives pointers on how to handle SPARC and Alpha systems. Like the man says, make sure you test this procedure before you use it. I'll add, don't try this on your production computer the first time. Use a sacrificial machine. But do it.

The next section is "Backing up Databases". The first chapter is an overview and general discussion. You can apply this to your MySQL or PostgreSQL installation. If you are running Informix, Oracle or Sybase, mine the appropriate chapter for useful information.

The final section, "Backup & Recovery Potpourri", covers ClearCase backup and recovery and miscellanea. In between those is a guided tour of backup hardware, which you should read when contemplating buying new hardware. However, you should also read the comments on media life and the care and feeding of backup media.

While Preston's experience is mostly in medium to large shops, and the book has a wealth of information for such shops, rest assured you can use this book even if all you have to back up is your own desktop computer. The basic concepts and techniques are the same regardless of the size of the shop.

Highly recommended. The job you save may be your own.

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

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

[Advanced search](#)

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

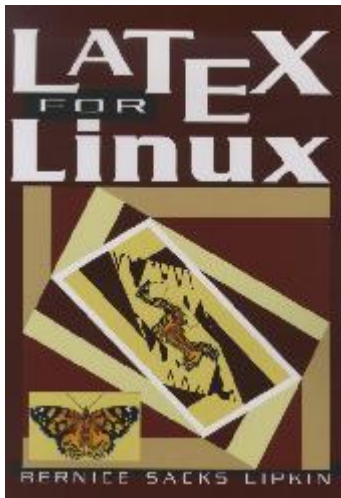
Advanced search

LaTeX for Linux

Ben Crowder

Issue #78, October 2000

For those of you scratching your heads wondering what all this TeX and LaTeX stuff is, read on.



- Author: Bernice Sacks Lipkin
- Publisher: Springer Verlag
- URL: <http://www.springer.co.uk/>
- Price: \$49.95 US
- ISBN: 0-3879-8708-8
- Reviewer: Ben Crowder

Anyone who has worked with Donald Knuth's TeX (pronounced "teck") or Leslie Lamport's LaTeX ("lay-teck") can tell you that both are extremely powerful but rather intimidating. LaTeX makes things easier but it's still hard for a beginner to grasp the connection between the marked-up text and the final, printed output. Enter Bernice Sacks Lipkin's *LaTeX for Linux*.

For those of you scratching your heads wondering what all this TeX and LaTeX stuff is, read on. Donald Knuth wrote TeX so he would be able to format his

book, *The Art of Computer Science*, the way he wanted to. Rather than focusing on the aesthetics of the printed output, Knuth wanted authors to concentrate on the content of the material. As such, TeX is not a word processor where you say, "The section heading should be larger than the other text, so I want font Becker at 14pt, and I want it centered." Instead, you tell TeX you want a section heading, and it automatically formats and numbers it for you. TeX is still low-level, so Leslie Lamport wrote LaTeX, a collection of higher-level TeX macros, to group some of the more common TeX commands together.

Lipkin's book is a well-written guide to the narrow, twisting path of learning LaTeX. First, I must say that the entire book was typeset using LaTeX, and as such, it provides an excellent real-life example of the various techniques within. This is a definite boon when you're trying to figure out exactly what a certain command does.

The first part of the book covers reading and understanding LaTeX. As with most programming languages, to be able to write, you must first be able to read. LaTeX is no exception. Chapter 1 covers what a LaTeX command does—the meaning of all those backslashes and curly braces. The second chapter defines concepts that TeX (and thus LaTeX) uses, such as commands, declarations, environments and scope. Chapter 3 explains LaTeX's document classes—letters, slides, articles, reports, books and so on—and how to use them.

Part II goes over things you'll need to prepare before you can start writing serious LaTeX. The fourth chapter gives you a practice template file to start working with. For die-hard Emacs fans, Chapter 5 helps you assign key bindings in Emacs to common LaTeX constructions. Since the whole point of LaTeX is to get your text into a printed (or otherwise viewable) form, the sixth chapter covers viewing and printing LaTeX files. You can convert LaTeX to HTML, text, PostScript, PDF, etc. When the inevitable errors happen, Chapter 7 will be there to help you deal with them.

The third part gets into the real meat of LaTeX. Chapter 8 covers the reserved single-character commands (such as the backslash). In the ninth chapter, we move up to single-word commands. Eventually, you'll get sick of typing the same instructions over and over again. Chapter 10 explains how to use macros and new commands to eliminate excess typing and frustration.

Part IV covers formatting text. While the default Computer Modern font LaTeX uses is nice, most people will want to use something different at times. Chapter 11 gives the nitty-gritty on everything you wanted to know about TeX's font system. When the time comes to write something in another language, you can learn from Chapter 12 how to place accents over and under characters. Like

HTML, TeX and LaTeX manage spaces for you—no matter how many spaces you put between two words, LaTeX will always put in the same amount of white space. While in most cases this is a good thing, at times you'll need to tweak the amount of space, and the thirteenth chapter explains how. Chapter 14 covers lists, and Chapter 15 explains how to align and indent text.

Figures, tables and marginal notes are called “floating objects” in TeX, since you don't specify an exact location for them. Instead, you tell LaTeX about where you want it to be (i.e., the top of the page), and it takes care of the rest. Chapter 16 explains how this possibly confusing feature works. Most academic writings are littered with footnotes, so the seventeenth chapter covers those. Chapter 18 describes the cross-referencing system, and Chapter 19 explains how to add comments (text that LaTeX will ignore, used mostly for writing notes to yourself) in your LaTeX code.

LaTeX is commonly used for typesetting math equations, and that's what Part V is all about. Chapter 20 goes over the math symbols and the grammar for getting into and out of math mode. Chapter 21 covers single-line math modes, and Chapter 22 explains arrays (the multiline math mode).

The next few sections of the book consist of only one or two chapters each. Part VI covers formatting in box mode; Part VII explains how to insert pictures and graphics into your text; and Part VIII shows you how to put the finishing touches (bibliographies, tables of contents and indices) on your document. Part IX describes how to design style sheets. The book ends with two appendices—one on the construction of a root file, and the other detailing how to convert from LaTeX to HTML and vice versa.

While visual tools like LyX are available to make writing in LaTeX less painful, many times you'll need at least a basic understanding of the language in order to get the desired result. This book will give you that understanding. It is well-written, concise, informative, and laced with the occasional bit of humor. Not many books are this thorough.



Ben Crowder has been heavily involved with computers for the past ten years, in almost every aspect (programming, graphics, networking, music and just about anything else you can think of). He has been working with Linux for the past two and a half years and has loved every second of it. In his spare time, he enjoys reading, writing, music and tweaking things on his Linux box. He currently lives in Utah and can be reached at mlcrowd@enol.com.

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

[Advanced search](#)

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

Advanced search

The XML Handbook, Second Edition

Daniel Lazenby

Issue #78, October 2000

For an authoritative reference on this flexible language and its possibilities, you cannot get better credentials than Charles Goldfarb and Paul Prescod.



- Authors: Charles F. Goldfarb and Paul Prescod
- Publisher: Prentice Hall
- URL: <http://www.phptr.com/>
- Price: \$44.99 US
- ISBN: 0-13-014714-1
- Reviewer: Daniel Lazenby

Some see XML as a new technology and books on how to use this new language regularly appear. In reality, it is not a new language, but one of those ten-year-old technologies that reappears on the scene as fresh and new. For an authoritative reference on this flexible language and its possibilities, you cannot get better credentials than Charles Goldfarb and Paul Prescod. Goldfarb is the inventor of the Standard Generalized Markup Language (SGML). Prescod was a member of the World Wide Web Consortium (W3C) that created the XML subset of the SGML standard. Besides the expertise of these two people, the book brings together the experience, projects and products of 27 national and

multinational corporations. The included CD-ROM contains a warehouse of trial and free software programs and related XML information.

The handbook is divided into 13 parts and 66 chapters. These 13 parts can be further organized into two groups, tutorials and sponsored chapters. Part I, Part XI and Part XII are XML tutorials. Industry experts from 27 corporations contributed chapters to the book, which are contained in parts II through X and organized by areas of interest. These chapters are clearly marked as sponsor-supplied. Part XIII is a guide to the CD-ROM and other XML-related books. There are too many chapters to list in this article. To see the table of contents, visit <http://www.phptr.com/> and search for the book title. The search will produce a web page containing the book's table of contents, a sample chapter, and a list of the corporate sponsors.

I found two paths through the book. The authors recommend reading Part I before going down either path. One path skips over the sponsor chapters and follows the tutorials. The other path wanders around, and within, the sponsor chapters. The tutorial path seems to support the more technical reader who is interested in learning to read and write proper XML documents. The sponsors' path seems to support the more business-oriented person who is interested in finding ways to enhance, automate, or webify their business processes.

The tutorials present the XML Language Version 1.0 and describe how to read/use XML-related specifications. The tutorials in Part I walk you through a high-level overview of XML's beginnings, the problem XML solves and major business areas benefitting from XML, as well as terms, definitions and the use and misuse of XML jargon. Part I tutorials are a prerequisite for the tutorials in Part XI, which focus on creating document-type definitions, creating XML documents, combining and sharing text between documents, formatting and validating a document's structure. One can easily jump from the tutorials of Part I to the tutorials of Part XI without hindering the book's overall flow or the reader's learning/exploration experience. For those with a thirst to know more XML details, there is Part XII, which demonstrates how to read seven XML-related specifications of the W3C. These tutorials require a thorough understanding of the material in Part XI and may be read in any order.

The sponsored chapters in Parts II through X cover several areas of interest by presenting a collection of application, tool and case study discussions. These discussions are grouped by: middle-tier servers, e-commerce, portals, publishing, content management, content acquisition, style sheets, navigation, and XML and Programming. This part of the book is not hands-on. It is more of a here is where XML has been used and what it has accomplished. Application discussions lean more toward the conceptual end of the spectrum. They offer high-level ideas about where XML has been (or may be) used to solve business

problems. Most of these discussions include a conceptual or architectural representation of the application being discussed. Tool discussions vary greatly between the various authors. Nevertheless, each author concentrates on what their tool can do with XML, rather than how to use the tool. As one might expect, case studies are a bit more concrete since they tend to examine real-world examples of what was set up. Often, they include snippets of XML used to create the page or function being presented.

The book concludes with a synopsis of free programs contained on the CD-ROM and in other XML-related books. The accompanying CD-ROM contains a virtual warehouse of trial and free software, white papers, specifications, markup and code samples, demos and sponsor product information. The 125 XML-centric software programs are not crippled, or time-limited, and are presented as free-use programs. Each of the 27 corporations is represented on the CD-ROM: some with just a link to their web site; others provide demos or information about their projects and products. Most of the CD-ROM product demonstrations require a Windows platform to perform properly. In some cases, there is "free" software for both Windows and Linux/UNIX platforms. Some free software includes source code. Links to a set of important XML web sites are also provided on the CD-ROM.

After I owned this book for a while, I begin to reread it from its index. I found the index to be robust, and it seems to contain an adequately detailed listing.

After reading this book, you will not have any programs or reusable code snippets. What you will have is knowledge of XML, a collection of XML tools and tool sources and a reliable XML Version 1.0 reference.

email: d.lazenby@worldnet.att.net

Daniel Lazenby (d.lazenby@worldnet.att.net) first encountered UNIX in 1983 and discovered Linux in 1994.

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

[Advanced search](#)

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

Advanced search

Securing Linux: Step by Step

Charles Curley

Issue #78, October 2000

The subtitle says it all: “A Survival Guide for Linux Security”.



- Author: Lee E. Brotzman, David A. Ranch, and a cast of 46
- Publisher: The SANS Institute
- URL: <http://www.sansstore.org/>
- ISBN: 0-9672992-0-9
- Price: \$49.00 US single copy; see web page for other options
- Reviewer: Charles Curley

The subtitle says it all: “A Survival Guide for Linux Security”.

This book is the result of an iterative process of consulting with experts in the field of computer and network security. The list of contributors includes staff at well-known organizations like the Computer Emergency Response Team (CERT) and the U.S. Census Bureau, so it is more than just the two main authors' expertise—it is a collaborative effort of 48 experts.

It is not simply a theoretical book on computer security. First, it details only one Linux distribution, Red Hat 6.0. Users of other distributions will be able to use the book as well, but they will need to fudge things according to the differences between their distribution and Red Hat 6.0. Users of Mandrake 6.x should have

no problem; users of Slackware will have to adjust a lot of the information on system startup. Debian users will probably find themselves scrambling to map all the RPM package names to Debian package equivalents.

Second, it is a step-by-step walk through the process. The authors don't simply say, "remove package foo"; they walk the reader through the process of removing package foo, with the complete command-line and system response for each command. It may be only one or two steps, but they are there to show you exactly what to type on the command line and what response to expect from the system.

The book is entirely command-line-oriented. This is good, in that the authors can show exactly what to do in each step. It also means you get to do a lot of typing and careful checking of your command lines. If you aren't already familiar with Bash's tab completion, now is a good time to read up on it in the man page.

Theory is minimal in this book. There is usually a brief discussion of each group of command-line steps. Then the steps to carry out are shown, interspersed with useful commentary.

The book is organized in a logical manner, starting with step one on security policies, the physical security of the computer, and a pre-installation check of the BIOS's security-related features (e.g., turn off the ability to boot from floppy). Each step is divided into sub-steps, so you can easily find an appropriate sub-step for any aspect of security.

Step two, which would be chapter two in any other book, deals with the installation of Linux. The authors cover pre-installation security, where they point out that (for example) an FTP installation from a public server on the Internet could leave your computer compromised before the installation is complete. Similarly, they discuss the security implications of partitioning.

It's no surprise that the authors prefer the custom installation of Red Hat over either workstation or server. Their motto is "When in doubt, leave it out", an excellent motto. If it isn't there, it can't be cracked. The installation step continues with password setup and some recommendations such as creating a boot diskette. The book then shows how to set system access policies and configure logging.

The next two chapters (excuse me—steps) are about securing a workstation on a network and a server on a network. The server step includes instructions for installing Secure SHell (SSH) tools, which are far more secure than the "r" analogs (rlogin, rsh, etc.), ftp or telnet. Other substeps show how to set up DNS,

electronic mail and several other services. The documentation on securing Apache includes password protection and adding `mod_ssl` to your Apache `mon`.

The process of securing a workstation includes disabling and removing a number of standard `mons`, or limiting access to those `mons`.

Step five deals with system tuning and packet firewalls. It gives a brief introduction to IPCHAINS, and shows how to make, install and test a strong ruleset.

Step six points the reader toward a number of tools for network security, such as the (in)famous SATAN and its descendants.

Appendix A has an excellent bibliography of Linux security resources on the Internet. Appendix B is the stock Red Hat 6.0 `/etc/inetd.conf`. Appendix C is a System V-style startup script for `ssh`, which fills a gap in at least two of the `ssh` products out there. Appendix D is a 20-page script for a strong firewall IPCHAINS ruleset, adapted for the book from David Ranch's highly respected Trinity OS.

Appendix E is a script to modify the permissions of a number of system utilities. The authors recommend you run it every time you install Linux. It is worth studying to see how insecure the authors find Linux to be.

The book is printed in an unusual format. It is spiral-bound, standard (North American) letter-size paper. The unusual part is that it is printed in landscape layout. The result is you see the book as a 17 x 11-inch sheet of paper, with the binding across the middle. This makes it possible to have a lot of information in front of you while working at the keyboard. There is plenty of white space for your notes. The effect was a bit disconcerting at first, but I found it easy to work with and rather like it.

The steps are wellwritten, and I was able to walk through several of the sub-steps. The only problems I had were caused by other problems in the system, ones outside the scope of the book. I was able to install `ssh`, for example, in minutes because the steps in this book are better than the README file that came with one of the distributions I tried.

One thing to keep in mind: while the book is a set of step-by-step instructions, you will have to remain alert to your own situation and local needs.

At first, I thought the scripts, especially the 20-page IPCHAINS ruleset, were not available on the Net. Well, I am glad to report that they are. The URL is carefully

hidden away at the beginning of Appendix A, which is not where the reader looking for, say, Appendix D is going to look.

I recommend this book to professionals in the field. If you are on the Internet with a firewall or any sort of server, you should read it and take the steps appropriate to your situation. As you do, check off each step completed so that you have a permanent record of how you have customized your firewall.

email: ccurley@trib.com

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

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

[Advanced search](#)

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

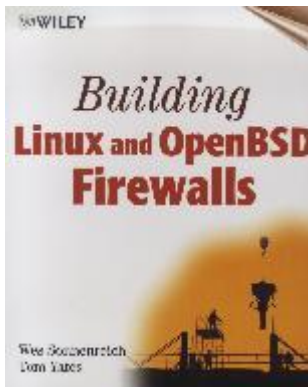
Advanced search

Building Linux and OpenBSD Firewalls

Ralph Krause

Issue #78, October 2000

Building Linux and OpenBSD Firewalls attempts to provide you with enough information to determine your security needs and create a firewall to meet them.



- Author: Wes Sonnenreich and Tom Yates
- Publisher: John Wiley & Sons
- E-Mail: info@wiley.com
- URL: <http://www.wiley.com/>
- ISBN: 4-713-5366-3
- Price: \$44.99 US
- Reviewer: Ralph Krause

More and more people remain connected to the Internet 24 hours a day, and it is no longer a question of whether they will be attacked, but when. The first line of defense against a break-in is a firewall, and an open OS such as Linux can be used to create a very secure one. While newer distributions try to make it easier to create firewalls, you still need to know some information so you can create one that can do its job well.

Building Linux and OpenBSD Firewalls attempts to provide you with enough information to determine your security needs and create a firewall to meet them. According to the introduction, it “is a cookbook for building firewalls using Red Hat Linux 6.0 and OpenBSD 2.5” and “contains step-by-step instructions on exactly how to build a very useful and powerful firewall from scratch”.

Even though it provides step-by-step instructions for creating and tuning a firewall, the authors believe your firewall will be more secure if you know what is being secured. To this end, the first three chapters cover basic network security issues. The first chapter discusses topics such as what you are protecting (your data, your computers and your reputation) and the value of good passwords. Chapter 2 provides a brief explanation of how the Internet works, covers protocols such as IP, TCP and UDP, and describes the common exploits against them. The third chapter explains some basic network configurations for a firewall and helps you determine which services should be provided by your network. These chapters also talk briefly about web browsers and Microsoft-specific problems such as Back Orifice.

The next two chapters are on choosing an OS and the hardware to use for your firewall. The authors provide a brief history of UNIX and free software, explain the differences between the GPL and BSD licenses, and offer comparisons between Red Hat Linux 6.0 and OpenBSD for such factors as software availability, ease of installation and general security. They also talk about building your firewall computer from the ground up so you know for sure what is in it and what to do when you have to open it up to fix something. They point out that you won't need bleeding-edge or high-performance hardware for the majority of your firewall situations, and provide some information on troubleshooting any hardware problems you might encounter.

Chapters 6 and 7 cover the installation of Red Hat Linux 6.0 and the steps you need to take to configure it as a firewall. The installation instructions are basically notes and enhancements to the Red Hat manual. The book then introduces ipchains and firewall rules and explains how to enable IP masquerading. A basic firewall script for ipchains is provided, along with instructions for starting up the firewall every time the computer boots.

Installing OpenBSD and configuring IPFilter are the subjects of the next two chapters. The book goes into more detail on installing OpenBSD than it does for Red Hat, including the creation of a boot floppy and hard-drive partitioning. Instructions on configuring the system to use your modem, mounting your CD-ROM and optimizing the kernel for firewalling are also given. The directions for configuring IPFilter follow the same path as the instructions on configuring ipchains, including an explanation of the book's basic firewall script and how to

start the firewall when the machine boots. OpenBSD tools such as IPNAT, IPFTEST, IPFSTAT and IPMON are also introduced in these chapters.

After explaining how to get your firewall up and running, the book moves on to the process of tuning it to be more effective. Specific firewall policies such as protecting against spoofed packets and blocking particular TCP services are given, with instructions on configuring both a Linux and an OpenBSD firewall to implement the policy. The book also explains how to determine what services your firewall is currently providing and how to shut down any you don't want to provide to the outside world.

Next, the authors cover intrusion detection and response. They discuss what to do during an attack, and how to evaluate the attack when it is over. They offer different scenarios for a home network, a network in a small business, and a large corporate network. This chapter also talks about monitoring your network, the importance of log files, and introduces tools such as SATAN and Tripwire to help you secure your network.

The final chapter is a hodge-podge of information. It includes notes on information from the earlier chapters, a brief introduction to the vi editor, and talks about the importance of having a security policy. Finally, it contains two small scripts: one to remove a disk set from OpenBSD and one to start your firewall under Red Hat Linux.

Building Linux and OpenBSD Firewalls covers quite a bit of ground in its twelve chapters. Almost one-half of the book is dedicated to Internet and network theory, but I still found this information relevant. After all, how can you build a secure firewall if you don't understand how it works and what it can protect? The authors attempt to make the subject matter easier to digest by using liberal doses of humor, although this occasionally makes the book hard to read. They also provide diagrams and sidebars to help explain complicated concepts. The authors provide a web site containing more Linux and OpenBSD scripts for firewalling, along with errata and updates for the book.

I found this book informative and useful. If you have a dedicated Internet connection or if you want to protect your small business from hackers, I think this book will help you.



Ralph Krause (rkrause@netperson.net) lives in Michigan and becomes more enthusiastic about open-source software each day. He is currently experimenting with various BSD flavors, as well as continuing to learn Linux.

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

[Advanced search](#)

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

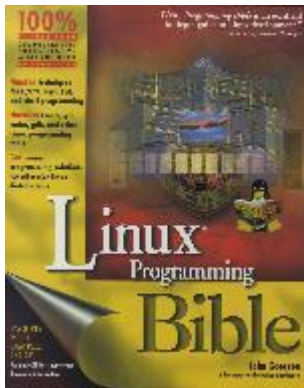
Advanced search

Linux Programming Bible

Ben Crowder

Issue #78, October 2000

This book shows both the forest and the trees of the exciting world of Linux programming.



- Author: John Goerzen
- Publisher: IDG Books Worldwide
- E-Mail: siteemail@idgbooks.com
- URL: <http://www.idgbooks.com/>
- Price: \$39.99 US
- ISBN: 0-764-54657-0
- Reviewer: Ben Crowder

As anyone who has spent some time with Linux knows, our favorite operating system is a programmer's heaven. With dozens of free compilers and interpreters for practically every language in existence, source code available for most programs and the programmer-oriented philosophy behind the system, Linux can easily be described as a programmer's dream come true.

Sometimes, however, because there's so much out there, it's hard to keep track of everything, especially for a newbie. That's where the *Linux Programming Bible* comes in. In clear, concise writing, it shows both the forest and the trees

of the exciting world of Linux programming. Note in advance that only a few topics are covered in deep detail; this book is more of a reference and tutorial for beginners, not a comprehensive mammoth of a book on one particular subject.

To start off the book, Chapter 1 introduces the Linux environment and how it relates to programming, including how to find documentation. The second chapter covers basic shell programming and is enough to get you started writing simple scripts which can serve as launchpads into more advanced shell programming. The third chapter, which could very well be one of the most important in the book, describes regular expressions in a fair amount of detail. It also explains how to use them in Perl, sed, awk and C/C++. Chapter 4 introduces the program that, in a sense, started the whole Open Source movement: Emacs. Finally, to conclude Part I, the fifth chapter covers the Linux file system layout, data files (such as /etc/passwd and /etc/group) and initialization scripts.

The second part of the book is devoted to the C/C++ environment. Chapter 6 gives an aerial view of gcc, with a few close fly-bys as well; it covers the different parts of the compiler, as well as some of the more commonly used options. The seventh chapter introduces a tool that has made many projects far easier to manage: GNU Make. It explains the basics of makefiles and goes through a handful of examples, which are enough to get you started. Chapter Eight covers memory management in Linux, statically and dynamically allocated memory, as well as some fun with those beloved pointers. The ninth chapter introduces both static and dynamic libraries and tools to help you manage the latter. When the inevitable bugs pop up, Chapter 10 is there to help you squash them—it explains how to use gdb, the GNU debugger.

Part III covers the Linux model, as the author describes it. The eleventh chapter explains the UNIX philosophy of “everything is a file”, and shows how to read and write from files. Chapter 12 introduces Linux processes: the basics, how to fork, synchronization and security-related items. Chapter 13 covers signals—handlers, sending/receiving and some dangers that programmers need to be aware of. Chapter 14 introduces the Linux I/O system, describing simple stream I/O, memory-mapped I/O, and **select()** and **poll()**. Chapter 15 closes Part III with an explanation of terminals and how to make sure your program acts nicely toward telnetters as well as toward those accessing it directly.

Nice programs talk with other programs, and that's what Part IV is about. Chapter 16 goes over shared memory and semaphores; Chapter 17 explains how to use pipes and FIFOs; and Chapter 18 introduces you to TCP/IP (Internet) sockets. Sockets are covered in more detail in Chapter 19.

Part V describes one of the most popular glue languages: Perl. Chapter 20 introduces the language, explaining the data structures and basic usage. Chapter 21 covers data manipulation—reading data, parsing and processing it, storing it and spitting it back out again. Perl has, in the eyes of many, been inseparably associated with CGI, and that's what Chapter 22 is about. Finally, Chapter 23 explains how to use Perl's database interface (DBI) to access SQL databases.

While command-line interfaces are quite useful and sufficient for many tasks, there are times when a graphical interface would be better suited. The sixth part of the book contains two chapters on the basics of X Windows from a programming viewpoint, Perl's (and Python's) graphical toolkit (Tk) and basic Gnome programming.

make made our lives easy. CVS made them even easier. The first chapter of Part VII explains the basics of CVS, including setting up your own repository, checking out and committing code, and branching/merging code. Chapter 27 describes some good security practices when writing code for Linux and Chapter 28 covers optimization. At the end of the book is a nice glossary that explains some of the more common terms used in the book.

Is it worth it? Unless you're a wizened kernel hacker with six or seven years of experience under your belt, the answer is yes. The book is a good overview of many topics, covers the most important subjects and can open your eyes to new, previously unseen horizons. This is a must-have book if you're interested in Linux programming.



Ben Crowder has been heavily involved with computers for the past ten years, in almost every aspect (programming, graphics, networking, music and just about anything else you can think of). He has been working with Linux for the past two and a half years and has loved every second of it. In his spare time, he enjoys reading, writing, music and tweaking things on his Linux box. He currently lives in Utah and can be reached at mlcrowd@enol.com.

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

[Advanced search](#)

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

Advanced search

Letters

Various

Issue #78, October 2000

Readers sound off.

Re: July 2000 Kernel Korner

In the July 2000 issue, Moshe Bar's article "Linux System Calls" contained information that I would like to correct. I suspect Moshe noticed by now that he was temporarily caught in a pre-i386 time warp when he stated, "This routine sets up an IDT (Interrupt Descriptor Table) with 256 entries, each 4 bytes long. It should be noted that the IDT contains vectors..." This is more a description of the pre-i386 IVT (interrupt vector table) than the IA32's IDT.

The IA32 uses another level of indirection. In particular, the IDT contains 8 byte descriptors which are either interrupt, trap or task gates. I suspect Linux uses the first two varieties. These gates contain the interrupt handler's 32-bit offset, some attribute bits and a 16-bit selector which references a code segment descriptor. In this case, that code segment descriptor is in the GDT (global descriptor table). This descriptor also consists of 8 bytes and vectors to the code segment containing the interrupt handler. The entry point for the handler is derived from the 32-bit offset lurking in the original IDT entry.

—Richard Sevenich rsevenic@netscape.net

Support for the WAP Article

I would like to express my support of more articles such as the one on WAP in the July issue. Recently, I was working on a WAP proof of concept for my organization. While I was motivated to move on the project, I was having difficulties taking the relatively new WAP standard and applying it in a straightforward manner to a commercial-quality proof of concept my organization would support. Mr. Mikal's article got to the meat of the issue and demonstrated how to get productive in the WAP space, while leveraging the skill sets that I, and other engineers here, already had. This one-page article got

the ball rolling on producing my organization's wireless presence. In doing so, another organization has embraced open-source and free software. It is articles like this, and the knowledge sharing of the Linux community, that make magazines such as *Linux Journal* welcome in my cubicle.

Thanks to Mr. Mikal and *Linux Journal*.

—Brad Wheat bradford.wheat@moai.com

iMac and Linux

When Linux began to grow in popularity, I jumped in, only to be put down by fellow Linux users. I used Linux on a Macintosh. As time went on and Linux distributions for the Mac grew, it became obvious that Linux users would have to acknowledge the Mac. But, rejection continued.

As a Mac user, I was aware of Linux before my PC pals. Apple had MKLinux, “an Apple thing”, I was told. Today, SuSE, the #1 distribution of Linux, is available for the PPC. I'll be merciful and not list all the others. After seeing the ads for PPC and Yellow Dog, it's nice to see a couple of articles, unlike the one degrading MKLinux in *LJ* a few months ago. Thank you.

—Calvin Bowen ctbowen@mindspring.com

The Last Python Letters (We Swear!)

Two things on your Python supplement:

First, the uproar over the naked man. I honestly didn't notice he was naked. After reading the Letters column, I went back and looked; and sure enough, nothing but a bowtie. Considering how much I've seen Laura Croft (and I've never seen or played whatever video game she's in), it only seems fair that some computer magazine would have a naked guy.

As for Guido's pet language (haha! ha! oh never mind...), I must admit I think it's a bit lacking compared to Perl in one way. A fixable way, mind you; this isn't a fatal flaw type of thing. The problem is cpan—Python seems to be missing it. Perl has cpan; pretty much all Perl modules can be found on cpan and installed with the following steps:

```
perl Makefile.PL
make
make test
sudo make install
```

Now, I've been looking at Python for a specific problem: I need to take data in a MySQL database and put it into a PostScript printer. Python seems to be able to

communicate with mySQL, and its piddle module is both delightfully named and PostScript capable.

However, I still can't get mySQL connections to work, and the piddle install file was bordering on useless (but I got it working). I'll keep at it because I have to, but I think the Python development community should attempt to recreate (and perhaps improve) cpan. As a developer, I've found cpan to be one of the best practical systems to enable code reuse. In fact, the C, C++ and pretty much all development communities would be wise to create their own versions of cpan.

—Kevin Lyda kevin@suberic.net

I just got through reading all of the letters in the July 2000 issue from people complaining about the cover of the supplement. Big deal! I own a computer store smack down in the middle of the bible belt. I put all of my old magazines out front for customers to look at while I am working on their computers, and not one person has complained about the cover. I service people from all walks of life, parents, teachers, clergy, children, etc., and no one has taken offense to the publication...in fact, it has caused Windows-only kind of people to ask about Linux. Have even had a couple of converts because of it.

Regardless of what anyone else says...keep up the good work.

—Rodney Rees rodney@crass-enterprises.com

I read the July 2000 *LJ* Letters, which was dedicated to reader responses to your Python supplement cover in May. I could well remember that supplement which motivated me to add Python to my language collection. It never occurred to me that the cover picture was offensive.

Mind you, I am a university lecturer, and my pigeonhole is transparent. This picture which annoyed so many in the Western world went here, in an Asian, predominantly Buddhist country, unnoticed!

—Visvanath Ratnaweera ratna@cs.pdn.ac.lk

Important Linux Sites

While flipping through the latest *LJ*, I saw a small section at the beginning, labeled "Important Linux Web Sites". What surprised me the most was to see <http://slashdot.org/> as the top site listed. This may be a result simply of a sort when putting the list together, but I think many readers would agree that slashdot.org is not a good representation of the Linux operating system or community. Opinions by both the maintainers and visitors of that site are

usually overly biased, ignorant or flame bait. If someone were to open an issue of *LJ* for the first time and wanted to learn more about Linux, I would much rather see them go to a more professionally run site, such as www.linux.com/ or www.gnu.org, than sites that sometimes make me embarrassed to be involved with Linux, like slashdot.org.

—Brian M Dial brian@flylife.org

Correction to “Linux on Wheels” Article

In the August issue of *LJ*, Linley Gwennap reported that the Delphi Automotive Systems Palm docking station (or MPC, short for Mobile Productivity Center) uses Windows CE. While WinCE plays a prominent role in Delphi's overall product strategy, it was not an appropriate choice for the MPC. Instead, we chose to use the eCos real-time kernel from Red Hat. I would appreciate it if you would publish this correction in the next issue of *LJ*.

—Brad Coon MPC Product Architect
Delphi Automotive Systems
bradley.s.coon@delphiauto.com

Internet Standards?

I was reading my fresh new copy of issue 75 of *LJ*. In particular, I was reading “Collecting RFCs” and found this first statement: “Requests for Comment (RFCs) are the standards of the Internet”.

It's not really true, because RFCs are what they claim to be, requests for comment from the Internet community. RFCs are the second stage of the life cycle of the “official” Internet community documents. They start as “drafts” proposed by someone. After a while, they become RFCs and then, after having received some comment from someone else, they become STD (Standard), FYI (For Your Information) or BCP (Best Current Practice). So, RFCs are not standard, but only a mature stage of a proposal for a standard.

You can find the same document with two different names, one for the last RFC status and one for the definitive status; e.g., we have RFC-2026 (modified from RFC-1602) and the BCP-9 that is titled “The Internet Standards Process—Revision 3”. Or STD-1 that collects (and obsoletes) RFCs 2500, 2400, 2300, 2200, 2000, 1920, 1880, 1800, 1780, 1720, 1610, 1600, 1540, 1500, 1410, 1360, 1280, 1250, 1200, 1140, 1130, 1100, 1083. For more information about this, try www.rfc-editor.org/rfc.html.

Anyway, I found your article very interesting and love *LJ*. Go on this way!

—Vincenzo Romano vromano@mail.com

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

[Advanced search](#)

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

[Advanced search](#)

UpFRONT

Various

Issue #78, October 2000

UPDATE TO THE APACHE SURGE...APACHE RULES, *LJ* INDEX and more.

EMBEDDED LINUX TSUNAMI HITS JAPAN!

An embedded Linux tsunami washed ashore in Tokyo on July 14th, as a handful of the world's most powerful electronics manufacturers including Fujitsu, Hitachi, Mitsubishi, NEC and Toshiba joined nineteen other companies and academic institutions to launch a Japanese embedded Linux consortium. The mission of the new Japan Embedded Linux Consortium (EMBLIX) strongly echoes that of the Embedded Linux Consortium (ELC) formed in Chicago earlier this year: to promote the use of Linux in a broad spectrum of next-generation intelligent devices and embedded systems. This strong showing of support from Japan's consumer electronics giants adds momentum to the already rapid proliferation of embedded Linux.

The founding members of EMBLIX include Advanced Data Controls, Access, Canon, CATS, Centura Embedded Systems, Densan, ERG, Elmic Systems, FDS Embedded Systems, Fujitsu, Gaio Technology, Hitachi, Lineo Japan, Metrowerks, Mitsubishi Electric, Montavista Software Japan, NEC Electronic Devices, Red Hat, Toshiba, Toyohashi University of Science & Technology, TurboLinux Japan, Waseda University and YDC. Dr. Tatsuo Nakajima (Waseda University) was named interim chair, while John Cheuck (TurboLinux Japan) and Y. Paul Kunimine (Gaio Technology) are serving as interim vice chairmen. First-year membership in EMBLIX costs \$1,000 per company and is waived for academic institutions. According to Cheuck, EMBLIX membership is restricted to corporations having a permanently established presence in Japan.

Visit EMBLIX at www.emblix.org.

Meanwhile, Back at the ELC...

Membership in the Embedded Linux Consortium surpassed 70 companies within three months of the group's March 1 launch. In June, the ELC elected its first Board of Directors: Dr. Inder Singh, chairman and CEO, LynuxWorks; Michael Tiemann, chief technology officer, Red Hat; James Ready, CEO, MontaVista Software; Tim Bird, chief technology officer, Lineo; Dan Bandera, business line manager, IBM Pervasive Computing; and Greg Wright, an independent Linux community member. Wright represents the 20+ "noncorporate" ELC members on the board.

Congratulations are also in order for Ralf Doewich, who won the ELC's logo contest. Doewich, whose entry placed first among a field of fifty, was the happy recipient of an HP C500 digital camera donated by ELC member Hewlett Packard.

Visit the ELC at www.embedded-linux.org.

Speaking of New Embedded Linux Orgs...

Organizers of a new Real-Time Linux Consortium (RTLX) will hold an organizational meeting during the Second Annual Real-Time Linux Workshop on November 29 in Orlando, Florida ("Tux Meets Donald Duck?"). The organizers have created a temporary RTLX web site where you can learn more about both the workshop and the proposed real-time consortium: www.thinkingnerds.com/projects/rtos-ws/rtlc.html

And Speaking of Conferences...

The second Embedded Linux Expo & Conference (ELEC) will occur on October 27 in Westborough, Massachusetts. The event combines an embedded Linux vendor expo with an all-day technical conference. The conference features technically oriented talks on integrating embedded Linux into information appliances, smart devices and other kinds of embedded systems. For further information, see www.rtcgroup.com/elinuexpo/index2.html.

Last, but Not Least...

The Embedded Linux market recently gained two new web resources:

- AllLinuxDevices: <http://www.alllinuxdevices.com/>
- SiliconPenguin: <http://www.siliconpenguin.com/>

Rick Lehrbaum (rick@linuxdevices.com) is founder and executive editor of ZDNet's LinuxDevices.com web site —"The Embedded Linux Portal".

UPDATE TO THE APACHE SURGE...APACHE RULES

In spite of Microsoft's advantage in marketing dollars, Apache continues to be the web server of choice. The issue came up again because Microsoft is attempting to migrate their hotmail.com site over to Windows 2000 boxes.

When Microsoft bought hotmail.com and linkexchange.com, they bought working sites based on FreeBSD. There were rumors of an earlier conversion attempt for hotmail.com, but apparently those were just rumors. Now it is happening.

The current report from Netcraft's web survey (www.netcraft.com/survey) Apache's market share and a decrease in Microsoft's market share. Thus, Microsoft seems to be going against the trend with <http://www.hotmail.com/>.

Here are some details from the survey:

Server		June 2000		July 2000		% Change	Apache		10,704,306		11,412,233	
+0.28	Microsoft	ISS		3,485,995		3,608,415		-0.50	Netscape-Enterprise		1,154,558	
1,225,085		1,225,085		+0.17								

Looking at longer-term trends, it was back in 1996 when Apache started getting significant market share and passed NCSA for the top slot. Over the years, Apache has experienced steady growth. Microsoft-IIS grew in market share up until the beginning of this year, but now continues to fall.

Why is this the case? We talked to one ISP in Canada and found that they run Apache on Linux, claiming to have about the same amount of traffic on their single machine as the ISP a block away has on their array of nine NT servers. Enough said.

MACS AND LINUX

Two months ago, our cover featured an iMac running Yellow Dog Linux. Next, I saw an iMac running SuSE 6.4. in the SuSE booth at the O'Reilly conference. What's happening?

Linux on the Mac isn't new. There has been Linux development for M68000 and PPC-based systems for many years. What has changed is both on the Mac and the Linux end.

First, the iMac has become a reasonably inexpensive platform with enough documentation on the architecture to make it a place to host your favorite OS. Today, for around \$1000, you can buy a cute little plastic box the size of a

monitor that has enough computing horsepower to run Linux. So, why not give it a try?

On the Linux side, what you get today is more appealing to the person who owns a Mac or might want to own a Mac. Two big issues are ease of installation and having a GUI-based system. In the August *LJ* review, installing Yellow Dog was covered. Clearly an easy install. While we haven't had a chance to install the SuSE version yet (the CD is on the way), if SuSE's recent install on Intel-based systems is any indication, it is going to be easy, perhaps very easy.

But, what about when the Mac user sees Linux come up? Will he be scared? Not likely. Yellow Dog defaults to the GNOME GUI, SuSE to KDE. Either is a reasonable desktop that shouldn't scare off the newcomer.

Finally, Mac on Linux (MOL), included with both distributions, allows you to access files and run native MacOS applications under Linux. So, if beige isn't the color you want for your Linux system, it looks like the time for colorful alternatives is here.

PARTY WITH *Linux Journal*!

A Friday the 13th party like you've never seen before....

LJ and the Atlanta Linux Showcase organizers are teaming up to host a grand party during ALS this October. (See www.linuxshowcase.org for show registration information.)

The evening will feature the first annual ALS Best of Show Awards and the Fifth Annual *Linux Journal* Readers' Choice Awards, followed by a fright-filled evening including music, dancing, prizes and lots of beer. But beware, the evening just may turn out to be more than you bargained for...

Stop by *Linux Journal's* ALS booth #417 to pick up your invitation.

ROCK LINUX

Rock Linux is a distribution whose claim to fame is that it's harder to install than other distributions. Rather than being user-friendly, this distribution tries to be "administrator-friendly"; that is, something an experienced UNIX sys admin would like. Its motto is minimalism: "All you ever wanted in a distribution—and less!" This distribution tries to get out of the way as much as possible, exposing you to the raw Linux system behind it.

Of all the major distributions, Rock Linux most closely resembles Slackware. Rather than using a custom packaging format like .rpm or .deb, Rock uses

ordinary tarballs as its packaging format. (*.tar.bz2, to be exact. The new .bz2 is a younger cousin of the ubiquitous .gz format. Its practical advantage is that bzip2 produces files approximately 20% smaller than gzip.) Like Slackware, Rock Linux prefers to patch upstream programs as little as possible. "If it's good enough for the upstream author, it's good enough for us!" Any desired customizations are the local sys admin's responsibility.

This is in sharp contrast to most distributions, which usually make many changes to the programs they include. They do this both in an attempt to "social engineer" the distribution to hide the operating system's complexities from the user, and to differentiate themselves from other distributions in the marketplace. Unfortunately, this social engineering comes at a price: inflexibility. Those snazzy GUI configuration dialogs may be nifty and easy to use, but if you need to change an option in a way the GUI designers didn't envision, you're out of luck. If you do try to outsmart the GUI tool and modify a text configuration file by hand, you may find that the GUI tool will happily overwrite your changes the next time somebody runs it.

Rock Linux has no GUI administration tools. However, there are a few command-line utilities provided to make the administrator's job easier. One is **runlvedit**, which helps you edit your system's run levels. (A run level specifies which daemons should be running in a particular situation. Thus, run level 2 might be your "normal" mode, run level 3 is without X, run level 4 is without the network, etc.) True to the Rock Linux philosophy, runlvedit doesn't invoke a dialog. Instead, it uses your favorite text editor as its user interface.

Rock Linux is not for the faint-hearted. You have to compile your distribution before installing it (!), and this will take days, even if nothing goes wrong—but of course, something will. The installation process consists less of choosing options in dialogs, and more of using standard Linux commands—mke2fs, mount, etc.—or changing a configuration file and then running a shell script to do the mundane work.

If you want to try and tame this beast, plan on spending a week or two to get familiar with it. You'll be rewarded with a more intimate knowledge of your Linux system and how it works than perhaps any other distribution can offer.

For more information on Rock Linux, see these URLs:

- <http://e-zine.nluug.nl/hold.html?cid=59> *Rock Linux: Not for woozies!* (a review)
- <http://e-zine.nluug.nl/hold.html?cid=1> *Rock Linux Philosophy* by Clifford Wolf, the author of *Rock Linux*
- <http://www.rocklinux.org/>, the distribution home page

THEY SAID IT

Why do you think they call it Red Hat?

“Linux sort of springs organically from the Earth. And it had, you know, the characteristics of communism that people love so very, very much about it. That is, it's free.”

—Steve Ballmer

“Happiness isn't something you experience, it's something you remember.”

—Oscar Levant

“Be courageous; it's the only place left uncrowded.”

—Anita Roddick

“It's not that we didn't like him as a spokesperson, we just liked him a whole lot more as a taco.”

—Conan O'Brien impersonating the Taco Bell decisionmakers.

“The future is no place to place your better days.”

—Dave Matthews Band

“Ways may someday be developed by which the government, without removing papers from secret drawers, can reproduce them in court, and by which it will be enabled to expose to a jury the most intimate occurrences of the home.”

—Lewis Brandeis, Supreme Court Justice, in a 1928 dissent that later became law

THE NEW RADIO

“It's personal. That's my entire philosophy of radio.”

—Larry Josephson

My favorite moment on “Saturday Night Live” was Paul Shaffer's impersonation of Don Kirshner, the helmet-haired, hyper-tanned record executive whose own syndicated rock concert often followed “SNL” on local NBC affiliates. In April 1978, Shaffer, playing Kirshner, introduced The Blues Brothers for the first time.

"No longer an authentic blues act", he said, The Blues Brothers had become "a viable commercial product."

Even then, Kirshner was an anachronism. It was already clear that the difference between *authentic* and *commercial* was the industrial heft required to make and move goods whose appeal was no less manufactured than the goods themselves. Commercial music and commercial broadcasting were two gears in the same machine: the business of stimulating and filling appetites for large quantities of music in the smallest possible varieties. "Saturday Night Live" was part of that machine as well. What else was on TV at that hour? Or ever?

Radio was hardly any better. Even the largest cities had fewer than a couple dozen signals that didn't fade within a few exits of downtown. For music stations, you could usually count the choices on one hand. Every one of those few stations tried to attract the largest numbers of listeners with the smallest varieties of tastes, so they could sell those numbers at top dollar to advertisers. The result was demand as homogeneous as supply, plus the ironic notion that the narrowest tastes comprised the broadest markets.

Also ironic was the belief that listeners comprised markets in any real sense. By the economics of commercial radio, *listeners* were the product, not programming. Stations and networks sold time to advertisers, not programming to listeners. Music and other programming was just bait. What listeners really wanted meant approximately nothing, which is what they paid for the goods. After all, they were consumers, not customers.

So can we blame them when they give music to each other at the same price?

Well sure. That's what the record industry has been doing by attacking Napster and wringing its hands over the "stealing" of copyrighted music that occurs when one music lover shares his or her MP3 collection with others over the Internet. They are joined by none other than our own leadership. Eric Raymond inveighs on behalf of artists' and record companies' right to distribute their property as they see fit, regardless of how easy it is for customers to share that property without anybody's consent. In a *Linux Journal* guest editorial, Eric writes, "The real point is that by 'sharing' without the artist's consent, *you deprive him of the right to control and dispose of his work*. The real question is this: are you going to support the artists, or steal away the few shreds of autonomy they might have left?"

Let's assume we answer "yes" to the first part of that question. The next question is, "Can we make a market where nothing is scarce and everybody can take what they want?"

Courtney Love points the way:

I'm looking for people to help connect me to more fans, because I believe fans will leave a tip based on the enjoyment and service I provide. I'm not scared of them getting a preview. It really is going to be a global village where a billion people have access to one artist and a billion people can leave a tip if they want to.

It's a radical democratization. Every artist has access to every fan and every fan has access to every artist, and the people who direct fans to those artists. People that give advice and technical value are the people we need. People crowding the distribution pipe and trying to ignore fans and artists have no value. This is a perfect system.

Is there a tip jar system out there already—or at least the beginnings of one?

Indeed, there is. We call it public broadcasting. Get past the noncommercial nature of the institutions that sell the service, and their often pathetic appeals for “support”, and you see the model at work. In some cases it works *better* than the advertising model, for the simple and efficient reason that the broadcaster's consumers and customers are the same people—a market grace advertising-supported broadcasters have never enjoyed (and paid subscription media like the one you are reading now *have* enjoyed, thank you very much).

Case in point. San Francisco's KJAZ was the longest-running commercial jazz station in the country until a few years ago, when its owner fell into deep debt and had to unload the property, which had been making money—not much, but enough to stay profitable. The owner made an appeal, and listeners sent in well over a million dollars in donations: more in a few weeks than the station made in a year from advertising, and none of it tax-deductible. It wasn't enough (the owner needed many more millions) and the donors got their money back; but further proof of the market model came when one of the local noncommercial stations picked up the format full-time, and reportedly made *more* money selling programming to listeners than KJAZ ever did selling time to advertisers.

I would gladly pay to play or hear anything I love. Right now, I pay to listen to four public radio stations and one public TV station. I don't see my purchases as “donations”, although I'm glad to claim the tax deduction. I see it as a tip jar system. And I would love to see some infrastructure built around it, so I could automatically make micropayments (or macropayments in a few special cases) for using artist-controlled material.

Why not set up a micropayment system for everything on TV, making TV a completely

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

[Advanced search](#)

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

Good-bye Bandits, Hello Security

Don Marti

Issue #78, October 2000

Don writes about the expiration of the RSA patent and the impact it will have on the Linux community.

This magazine should be coming out in September, right about the time that RSA Inc.'s US Patent number 4,405,829 expires. If you get this before September 20, be sure to visit the "RSA Freedom Clock" web site (see Resources) where you can count down the days, hours, minutes and seconds. (There's no legal precedent saying exactly what time a patent expires, so wait a day or two after the clock runs down so you don't risk infringing it.)

What does the RSA patent cover? A lot of security-sensitive software that uses public-key encryption, including both the Secure Socket Layer implementations in commonly used web browsers and servers, and version 1 of the SSH protocol, as commonly used for secure remote administration. As I write this, Linux distributions still can't legally distribute a lot of useful, interoperable security software. That will change when the patent expires, and my friends and I are going to have a big party somewhere in the Silicon Valley area to celebrate. Check your local Linux user group for the patent expiration party in your area.

Over the last year or so, we've heard a lot about how software patents, which became legal in the U.S. in 1981, stifle innovation. James Bessen and Eric Maskin conducted a study showing that "far from unleashing a flurry of new innovative activity, these stronger property rights ushered in a period of stagnant, if not declining R&D among those industries and firms that patented most". And a legal minefield of intersecting patents threatens all software development, free or proprietary. Where the software patent mess intersects with important basic security work such as encryption, software patents interfere with the spread of best practices and can be a major security problem.

Software patents hurt their holders, too, by skewing their view of reality. Somehow, these patent holders assume that just because people are giving them money, those people must like them. Imagine a group of bandits who camp around a well in a desert and begin charging travelers for water. Real bandits would keep their weapons close by at night. But today's patent bandits seem to think that people want to do business with them.

Unisys, for example, owns a software patent on a compression algorithm called LZW, which is used in the Web's obsolete but common GIF file format. Unisys's web site carries the preposterous claim that "demand for LZW-related technology by web developers continues to grow", when web developers are simply using GIF for compatibility with old browsers that don't support free formats such as PNG. There's no love for Unisys here. Like the bandits around the well, Unisys is getting paid because the travelers have no choice for now. But unlike the bandits, Unisys seems to think that the travelers are coming to them because they want to. When enough of the browser-installed base supports PNG, Unisys can say good-bye to their royalties, and the Web can say good riddance to a pretty dumb bunch of bandits.

RSA Security, Inc. is another set of bandits who are going to get a surprise right about now. Behind all their fluffy talk of "partners" and "solutions", they're just charging people to draw from a well that U.S. taxpayers helped dig when we funded crypto research at MIT. As a nation, we give educational institutions special tax treatment. This isn't just because they grade our papers and answer our questions during office hours. Educational institutions are about research, too, about extending the scope of useful human knowledge. Doing research as a non-profit and patenting the good parts for profit is a big ripoff any way you look at it. When the IRS finally starts raiding the big research "universities" such as MIT and google.com's little tax shelter, Stanford, I'll throw another party.

With a little scrutiny, and hopefully some long-awaited software patent reform, patent bandits will start to fade away as a security threat. The U.S. government's National Institute for Standards and Technology is selecting an algorithm as the new national Advanced Encryption Standard, and they're alert enough to not let the next RSA or Unisys seize control of the well. The AES web site says, "NIST hereby gives public notice that it may seek redress under the antitrust laws of the United States against any party in the future who might seek to exercise patent rights against any user of AES that have not been disclosed to NIST in response to this request for information". Which is basically bureaucrat-speak for "try to run a patent burn on us, and we will go John Woo on your sorry ass".

Thank you, NIST. But the next great threat to Linux security comes not from corporate software patentees or government agencies seeking "IP Wiretaps

Everywhere". Since earlier this year, the bad old U.S. crypto export regulations are effectively gone if you make your source freely available. The threat comes from within. If you do a Linux distribution, pay attention. After this month, there is no excuse not to supply the best industrial-strength crypto available. So there's no excuse not to make security a design consideration instead of a sheepish warning after a distribution ships with a hole. Free clue with the purchase of every magazine: read the OpenBSD security page, and do that. If you can't afford to be in the security business, you can't afford to be in the Linux business.

At *Linux Journal*, we expect to see OpenSSL and OpenSSH well-integrated into the distributions. That's the easy part. Here's the hard part: don't load users up with potentially exploitable services. Please, don't let Marketing talk you into shipping Linux with a loaded gun duct-taped to the user's foot, just so you can put one more checklist feature on the package. The person who wants to install Linux at his or her company gets set back in the struggle every time someone publishes a security flaw in SquirrelTech Linux's network acorn-balancing d\#\#230>mon. I don't care what all nifty-keen new software you offer—any user who wants his box to be a virtual food court of services should ask for it. The Bastille Linux project is an important step in the right direction, but its very existence shows that the mainstream Linux distributions are thinking about security about as much as a hog thinks about Feynman diagrams. The users who need security most—the beginners—aren't getting it right out of the box.

The world needs security-aware network administrators now more than ever. But a user shouldn't have to be one just to run Linux. If you ship a distribution with an installation designed for Joe Six-Pack, and a security posture so wide open that you need to be Rick Moen to lock down, then something is terribly, terribly wrong. And you don't have an excuse any more. "We can't ship crypto." Wrong. See above. "Users expect standard services." Wrong. Maybe UNIX-using graduate students do; regular people expect security. "Don't worry, it'll be behind the firewall." Wrong. It's hanging out buck naked on a DSL connection. Or maybe it *is* the firewall. "If we change it, it'll be harder to support." Wrong. How easy is it to "support" a root compromise? Thanks to the RSA patent's expiration, the time for security excuses is over. Don't let marketing weenies make security policy, and you'll be issuing a lot fewer security advisories.

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](#)

UnixWare and Linux Get Hitched

Phil Hughes

Issue #78, October 2000

Having UNIX and Linux coming from the same company will also make it much easier to get applications written, ported and supported in both environments.

Remember Xenix? That is what Microsoft called their UNIX-derived OS, before they pawned the PC version off on Santa Cruz Operation (SCO) in 1984. SCO, which had its own version of UNIX, also bought UnixWare from Novell in late 1995. UnixWare was Novell's name for the original AT&T UNIX, which Novell bought a couple years earlier for more than \$100 million in Novell stock.

Some Novell engineers who worked on UnixWare are currently at Caldera. And now they're getting it back. In early August, Caldera acquired the Server Software and Professional Services divisions of SCO. This now gives Caldera several kinds of UNIX to sell (including Linux and UnixWare), while SCO gets a 28 percent stake in Caldera in the form of 17.5 million shares of Caldera stock.

How things have changed. In 1996, SCO sent spam faxes telling people they could get a \$50 trade-in on their copy of Linux by upgrading to SCO UNIX. In October 1996, I asked Bryan Sparks, then CEO of Caldera, about SCO. Specifically, I asked whether Caldera would have a market if SCO had embraced Linux. At the time, SCO was very big in the OEM/VAR market, which was where Caldera was headed. Bryan agreed with my theory, pointing out that many of Caldera's IVPs (Independent Vendor Partners) were from the SCO camp. (That interview was published in the January 1997 issue of *Linux Journal*.) Now those camps will be in one company: Caldera.

This is good for Linux. With one company offering a choice—UNIX or Linux—the customer has to take the most useful and market-ready alternative. Those of us who have been around for a while know that in most cases, Linux outperforms SCO UNIX. But making that choice isn't a no-brainer for companies that also have to switch vendors and support organizations. Caldera's SCO acquisition takes care of that problem.

Having UNIX and Linux coming from the same company will also make it much easier to get applications written, ported and supported in both environments. SCO's huge existing support structure is a big win here.

Finally, SCO, Sequent and IBM have been working together on the Monterey project, which aims to deliver a single UNIX product line spanning the Intel IA-32, IA-64 and Power PC platforms. Having all this cooperation under one roof—a Linux-based roof—is going to make it much easier to go head-to-head with NT for server solutions. Given a mix of Apache's current market penetration (62.53%, according to Netcraft), SCO's global marketing and support infrastructure and Linux, Caldera should give Microsoft a run for its money in the server and e-business arena.

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

[Advanced search](#)

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

Best of Technical Support

Various

Issue #78, October 2000

Our experts answer your technical questions.

Dependencies and Libraries

Recently, I downloaded a Linux application and when I tried installing it, I encountered an “unsatisfied dependencies” error message in the package manager. Where do you go for missing Linux libraries and how do you install them? —Kelvin L. Barnes, Kelvin.Barnes@worldnet.att.net

If you're using an RPM-based distribution such as Red Hat, you install missing libraries the same way as any other software (i.e., by installing RPMs with the rpm program). You can find the RPMs containing these libraries by searching <http://www.rpmfind.net/> or <http://www.rpmsearch.net/>. On my Red Hat 6.1 box, I've installed the rpmfind utility (which was on the CD) provided by these sites, and it makes finding and downloading RPMs a snap:

```
rpmfind -v --auto --latest  
your_rpm_name_here
```

You still have to install the RPMs (as root, typically) after they're downloaded:

```
rpm -Uhv
```

Another place to search is your distribution's download sites, which sometimes include software that wasn't shipped on the CD for various reasons. Red Hat's sites (use mirror sites if possible) are at <ftp://ftp.redhat.com/> and <ftp://updates.redhat.com/>. Occasionally, RPMs designed for one distribution may not operate correctly on another—the RPM itself is okay, but the files contained in it may have been configured differently, or they may be placed in different locations than normal for your distribution. If given a choice, choose an RPM that matches your distribution (e.g., if you're running Red Hat and you see “redhat” in the URL of one of the RPMs, that's probably the one you should choose). I have rarely encountered this problem, though. —Scott Maxwell,

maxwell@ScottMaxwell.org Was this “application” a RPM file? If so, it should have taken care of all dependencies. Otherwise, the “missing” libraries could be an infinite number, so reading the application's documentation, specifically about requirements, should point you in the right direction. Usually your distribution's web site should have all sources for your Linux libraries and you should be able to download almost (99%) everything from there, unless it is a very specialized library. —Felipe E. Barousse Boué, fbarousse@piensa.com

One Working Parallel Port

I've installed two additional parallel ports on a PC, and I would like to use them all. So far, only /dev/lp0 works. Echoing into /dev/lp1 and /dev/lp2 returns the following nastygram:

```
BASH: /dev/<device name>: Device not configured
```

The ports work under Win98, so there are no IRQ or I/O address conflicts. Here's some more information:m

IRQ	I/O range	Win98 name
7	0378-037f	lpt2
5	03bc-03be	lpt1
9	0278-027a	lpt3

—Luis F. Gonzalez, gonzalezl@saic.com

Using ports in Linux requires two steps. First, your kernel must recognize the devices. You can check this by either scrolling through your boot messages by typing **dmesg | less**, or checking /proc/ioports and /proc/devices. The second step is to create the device file in /dev. If this file does not exist, you can use the **makedev** script found in the same directory to create most Linux device nodes. If it does exist, your problem is most likely with the first step. —Chad Robinson, crobinson@rfgonline.com If you've installed the kernel sources, the file /usr/src/linux/Documentation/parport.txt addresses this in detail. Briefly, if the ports are not automatically recognized and configured, and if printing and parallel port support are compiled as modules (not compiled into the kernel proper), try adding the following lines to the file /etc/conf.modules:

```
alias parport_lowlevel parport_pc
options parport_pc io=0x3bc,0x378,0x278 irq=5,7,9
options lp parport=auto
```

Then, from the shell, try removing and reinserting the relevant modules (the order is important):

```
rmmod lp
rmmod parport_probe
rmmod parport_pc
rmmod parport
insmod parport
```

```
insmod parport_pc
insmod parport_probe
insmod lp
```

You will need to do all of this as root. I'm not an expert at configuring parallel ports, since it normally works correctly out of the box for me; I've done it on only one machine, and on a different distribution from yours, so I may have overlooked something. You may need to experiment a little before it works. Perhaps some ambitious reader will turn parport.txt into a proper HOWTO, if that hasn't been done already. —Scott Maxwell, maxwell@ScottMaxwell.org

Booting Problem

I have two hard drives on my system. One is dedicated to Linux, the other is for Windows 98 (second edition). I can't boot to Win98 when both drives are up. They're set up fine (one master, one slave, BIOS finds them, etc.). I can boot Linux with both drives connected. If I disconnect the Linux drive, I can run Win98 fine. LILO seems to know that it has a DOS drive and a Linux drive. If I say "dos" at the boot prompt, LILO doesn't complain—things just hang. I've even tried booting with a Win98 boot floppy; the system reads from the floppy for a while and then just hangs—no message, no nothing. One drive is a WD, the other is IBM. Please help! —Mitch, mitchp20@home.com

I am assuming that the master disk has Linux on it, therefore you can boot fine with both disks active. It seems there is just a misconfiguration of your lilo.conf file. You have to tell LILO where (which physical disk and which partition) DOS/Windows is located. Check on the specified devices in /etc/lilo.conf. Look in the section where you have your "dos" label for /dev/hdaX and /dev/hdbY (X and Y being partition numbers) and hda, hdb representing your physical disks. You may need to find which disk holds what. Two lines like:

```
other=/dev/hdb1
label=dos
```

may be what is needed for you to be able to boot DOS (and Linux) with both disks on. Don't forget to write that LILO info to the boot sector with **/sbin/lilo -v** after you have made changes in /etc/lilo.conf. —Felipe E. Barousse Boué, fbarousse@piensa.com

You don't say whether Windows is on the slave or the master. By default, Windows won't boot from a secondary drive. However, you can try booting from the second drive by putting the following in lilo.conf to boot DOS/Windows (it swaps the drive letters):

```
other = /dev/hdb1
label = dosb
loader = /boot/any_d.b
```

—Marc Merlin, marc_bts@valinux.com

Function Keys Not Working

I am running a console on an HpUX box. This application is run from an xterm on my Linux box. The HpUX box has no X libs and does not accept my function keys. Does anyone know how to make my function keys work while telneted into this HP system? —David, uxidlm@netscape.net

Since you state that the HpUX has no X libs, and it looks to me that you are just using a terminal emulation window (xterm) in your Linux box, you should be able to define which code to generate whenever you press a function key so it gets sent to the HpUX box through your xterm. You can use the **xmodmap** command to define such sequences on your Linux box. Put those sequences in the `.xmodmaprc` file of your login account or in Xmodmap's general configuration file. Try also keying ESC-1 for F1, ESC-2 for F2, etc., to test whether your application is getting Fkey sequences. Lastly, terminal emulation affects the way Fkeys work. You should have the terminal emulation correctly set up, with the termcap definitions in both the HpUX side and the Linux xterm session. --Felipe E. Barousse Bouéfbarousse@piensa.com I don't have experience with this specific case, but the normal way to get your function keys (and other nonstandard keys) recognized is to modify your TERM setting after logging in on the remote machine. You do this with something like **export TERM=termttype** (for bash or ksh) or **setenv TERM termttype** (for csh or tcsh). You may need to experiment with values for "termttype"--I would try values like "xterm", "vt100", "vt220", "hpterm" and "linux" (in increasing order of desperation). If none of those works, look through `/etc/termcap` on the remote machine for other possible values of TERM. When you find a value that works, simply copy the TERM assignment into your shell startup file (`.bash_profile`, `.cshrc`, or whatever) on the remote system, and you'll never have to worry about it again. —Scott Maxwell, maxwell@ScottMaxwell.org

Cannot Access C: Drive

I lost the link to my C: drive (Windows). In the process of trying to connect my scanner, I changed sda1 to hda1 and now I can't gain access to my C: drive from Linux or from LILO. When it gets to the LILO prompt and I select Windows, it just locks up. How do I get the link back? —Faron Ducharme, faron_ducharme@ti.com

You don't mention how you "changed sda1 to hda1". However, LILO's configuration information is stored in `/etc/lilo.conf`. After editing this file, you will need to run LILO from a command prompt to tell it to reread its settings. — Chad Robinson, crobinson@rfgonline.com Does Linux boot fine? Seems you are using SCSI disks, so if you renamed sda1 to hda1 in the `/dev` directory, changing

it back should solve the problem, providing you haven't changed LILO's configuration at `/etc/lilo.conf`. Logged in as root, use `mv /dev/hda1 /dev/sda1`. If you cannot boot Linux, then you would have to use the emergency boot disk (you have one, don't you?) you made at installation time to be able to boot the machine and then change the device names back to normal as indicated above.
—Felipe E. Barousse Boué fbarousse@piensa.com

[4216s1.html](#)

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

[Advanced search](#)

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

[Advanced search](#)

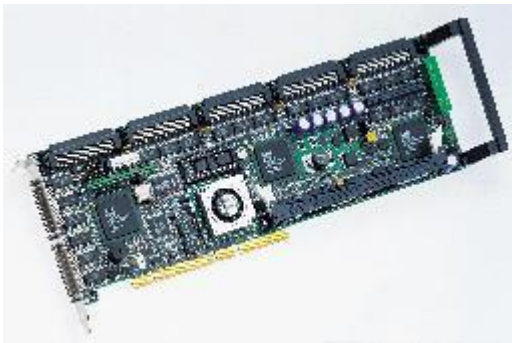
New Products

Ellen M. Dahl

Issue #78, October 2000

64-Bit PCI-Wide/Ultra160 SCSI, ClusterWorX v1.2 and more.

64-Bit PCI-Wide/Ultra160 SCSI



The 64-bit PCI-Wide/Ultra160 SCSI RAID controllers with optional clustering support are available from ICP vortex. These controllers join ICP's powerful RN series, supporting RAID levels 0, 1, 4, 5 and 10. The ICP GDT7523RN, GDT7543RN, GDT7563RN models offer 2, 4 or 6 Ultra160 channels and support data transfer rates of 160MB/sec per channel, with up to 15 devices attached to each channel on cables up to 12 meters in length. They feature 64-bit architecture for high data transfer rates. The controllers are PCI 2.2 compliant with full bus mastering capabilities. They can be equipped with one 16MB, 32MB, 64MB or 128MB standard unbuffered PC100 DIMM module, with or without ECC. Other features include automatic cache RAM detection, multi-level delayed write and read-ahead cache with intelligent self-adjusting management. ICP controllers support Linux and are all intelligent input/output-ready.

Contact: ICP Vortex Corporation, 4001 E. Broadway, B-20, Phoenix, AZ 85040, 602-414-0414, 602-414-0444 (fax), sales@icp-vortex.com, <http://www.icp-vortex.com/>.

ClusterWorX v1.2

ClusterWorX version 1.2 provides users with a GUI featuring both command line and HTML. Management tools include remote access, disk cloning and serial access to nodes, including remote monitoring and resetting of individual nodes without affecting the uptime of the entire system. Features are architecture-independent with support for any standard ATX or XATX motherboard. ClusterWorX software is distributed with Linux NetworX cluster systems and is not yet offered as a separate software package.

Contact: Linux NetworX, 8689 South 700 West, Sandy, UT 84070, 801-562-1010, 801-568-1010 (fax), sales@linuxnetworx.com, <http://www.linuxnetworx.com/>.

FacetTerm for Linux

FacetTerm provides multiple sessions on PCs with terminal emulation and on character terminals. The multi-session and productivity software works with all character-based UNIX applications (without change) and all text terminals or PC terminal emulators. FacetTerm Version 3 is compatible with Red Hat, SuSE, Caldera OpenLinux and TurboLinux, and is supported on all other versions of Linux with the glibc version 2 runtime libraries.

Contact: FacetCorp, 1820 Preston Park Blvd., Suite 1000, Plano, TX 75093, 972-985-9901, 972-612-2035 (fax), info@facetcorp.com, <http://www.facetcorp.com/>.

LifeKeeper for Linux 3.0

SteelEye Technology Inc. announced its LifeKeeper for Linux 3.0, an automated, high-availability clustering solution, commercially available on Red Hat Linux, that ensures continuous availability of business and mission-critical applications and data as well as 24 x 7 system uptime. LifeKeeper is suited for web-centric enterprise applications as it provides IT and system managers with the ability to implement fail-over and system back-up solutions that support round-the-clock availability of applications, data and Linux servers in enterprise e-commerce environments.

Contact: SteelEye Technology, 2660 Marine Way, Suite 200, Mountain View, CA 94043, 650-318-0108, 650-938-4291 (fax), info@steeleye.com, <http://www.steeleye.com/>.

MachZ PC-on-a-Chip

The ultra-low-power MachZ PC-on-a-Chip is available from ZFLinux Devices. MachZ Integrated Development Systems include Red Hat Linux 6.2,

LynuxWorks BlueCat Linux and LynuxWorks development tools for creating compact, customized Linux embedded applications. The MachZ consumes less than two watts of power and runs at temperatures low enough to make cooling fans unnecessary. The software suite integrates a Phoenix BIOS and customized Linux implementation or VxWorks RTOS tailored to the MachZ's hardware features, including PCI, ISA and Access (I2C) buses, serial and parallel I/O, floppy and hard disk controllers. The chip is a .25 micron technology device in a 388BGA, 35mm by 35mm package.

Contact: ZFLinux Devices, Inc., 1052 Elwell Court, Palo Alto, CA 94303, 650-965-3800, 650-965-4050 (fax), info@zflinux.com, <http://www.zflinux.com/>.

MediaXpress

MediaXpress is a knowledge and content management solution allowing delivery of external news to a larger management system as an easy "snap in" to most any system. The corporate portal solution utilizes powerful XML technology to sort, customize and deliver real-time news and other digital content from over 400 media sources directly to a customer's web server.

Contact: WAVO Corporation, 3131 East Camelback Road, Suite 320, Phoenix, AZ 85016, 877-834-6397 (toll-free), 602-468-5790 (fax), salesinfo@mediapress.com, <http://www.mediapress.com/>.

Gateway Guardian

Gateway Guardian offers an affordable, advanced security software solution. The Linux-based firewall products are available in Personal and VPN editions, designed to protect networked computers with a quick software install on a PC, and also in a Professional edition.

Contact: NetMaster Networking Solutions, Inc., 46165 Yale Rd., #317, Chilliwack, BC V2P 2P2, Canada, 888-335-3638 (toll-free), 604-792-0911 (fax), info@netmaster.com, <http://www.netmaster.com/>.

NETspider-U PCI 128K ISDN Card



Traverse Technologies released the NETspider-U ISDN card for Linux. The NETspider is a fully integrated internal modem card providing high-speed ISDN access to the Internet, intranets and on-line services. It eliminates the need for an external ISDN router/modem, and allows a Linux machine to be a router/firewall/gateway with connectivity at 64, 128 or 256KBps. Benefits are its fast and efficient HTML sessions; downloads and file transfers supporting PPP, ML-PPP and RAW IP protocols; support of the US NI-1 ISDN switch protocol; easy installation and low cost. NI-1 HiSax drivers are available for Linux 2.0.36 and 2.2.x kernels.

Contact: Traverse Technologies, 652 Smith St., Clifton Hill, Victoria 3068, Australia, +61-3-9486-7775, +61-3-9482-7754 (fax), info@traverse.com.au, <http://www.traverse.com.au/>.

BestAcct Accounting Software

BestAcct is a suite of corporate business accounting programs, designed exclusively for the Linux desktop. BestAcct meets rigorous financial accounting standards and includes the following modules: General Ledger, Accounts Receivable, Accounts Payable, One-Step Check-Writer and Sales Invoicing/Receipts. A free evaluation copy of the programs may be downloaded from the web site.

Contact: Proven Software, Inc., info@BestAcct.com. <http://www.BestAcct.com/>.

qCF for Linux

Quadron's qCF for Linux is a software development toolkit for ARTIC communication cards installed in Linux servers or workstations. It enables users to create an exact solution for their communication requirements. They can run async, bisync, HDLC or custom variations on a Linux/ARTIC system. The ARTIC card has its own on-board processor, making it a dedicated programmable communications computer that resides within the Linux computer system. The toolkit comes with a real-time analysis tool for debugging and gathering performance data, as well as communication statistics gathering facilities for error detection and correction. Three PCI-bus ARTIC cards are available for qCF for Linux: ARTIC186 8-Port PCI Adapter, ARTIC186 Model II ISA/PCI Adapter and ARTIC X.25 ISA/PCI Adapter.

Contact: Quadron Corporation, 209 East Victoria St., Santa Barbara, CA 93101, 805-966-6424, 805-966-7630 (fax), info@quadron.com, <http://www.quadron.com/>.

SMP for PowerPC

Synergy Microsystems announced support for Linux with Symmetric Multiprocessing (SMP) on all its PowerPC-based multiprocessor boards. This is the first release of Linux SMP for VME and CompactPCI PowerPC boards and includes Yellow Dog Linux Champion Server 1.2. Linux SMP runs on all of Synergy's multiprocessor PowerPC G3/G4 boards, including the dual-CPU VGM5 and quad-CPU VSS4 for VME, and the dual-CPU KGM5 for CompactPCI. The OS is fully compatible with the Synergy Scientific Subroutine Library (SSSL).

Contact: Synergy Microsystems, Inc., 9605 Scranton Rd. Ste. 700, San Diego, CA 92121, 888-479-6374 (toll-free), 858-452-0060 (fax), sales@synergymicro.com, www.synergymicro.com/linux.html.

tRAID

tRAID provides users with a scalable and robust enterprise-class storage solution at an open-systems price. The BigStorage tRAID can be configured to hold over two terabytes of usable storage per subsystem, and it can chain two additional 3U JBOD units to the controller unit. Available in both UltraSCSI and FibreChannel versions, the tRAID offers reliability with top-notch controller design, hot-swap SCA drives and power supplies, and high-airflow layout. The tRAID is backed by BigStorage's 24/7 Live Response customer support system to guarantee maximum client uptime.

Contact: BigStorage Inc., 19 Heron St., San Francisco, CA 94103, 800-846-3789 (toll-free), 415-252-1521 (fax), info@bigstorage.com, <http://www.bigstorage.com/>.

Digital Image Processing



Wolfram Research announced the release of Digital Image Processing, a new Mathematica application package with a collection of fundamental and advanced image processing tools. Its functionality is integrated into Mathematica, allowing users to perform much more sophisticated analyses with it than with standard image processing software. With Digital Image Processing, one can import and export more than 15 standard image formats;

use grayscale, RGB, HSV, and CMYK color models; apply dozens of predefined filters and filter-design algorithms; perform region-of-interest processing over arbitrary polygonal regions; and smooth and sharpen images and reduce noise. Digital Image Processing is designed for use with Mathematica 4 or later versions and is available for several platforms, including Linux (PC).

Contact: Wolfram Research, Inc., 100 Trade Center Dr., Champaign, IL 61820-7237, 800-965-3726 (toll-free), 217-398-0747 (fax), info@wolfram.com, www.wolfram.com/products/applications/digitalimage.

e-smith 4.0

e-smith announced the newest version of their Linux-based communications server for small and growing businesses. e-smith server and gateway 4.0 offer reliability and simplicity in e-mail, web access and other essential on-premise, network-based services at a fraction of the cost of other alternatives. Based on the latest version of Red Hat Linux, the e-smith server and gateway automates installation and integrates reliable open-source technology such as Apache Web Server, Qmail Mail Transport Agent, ProFTPd FTP Server and Squid Internet Object Cache. The system can be administered from any desktop using a standard web browser.

Contact: e-smith, inc., 150 Metcalfe St., Suite 1500, Ottawa, ON K2P 1P1, Canada, 613-564-8000, 613-564-7739 (fax), sales@e-smith.net, <http://www.e-smith.com/>.

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

[Advanced search](#)

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