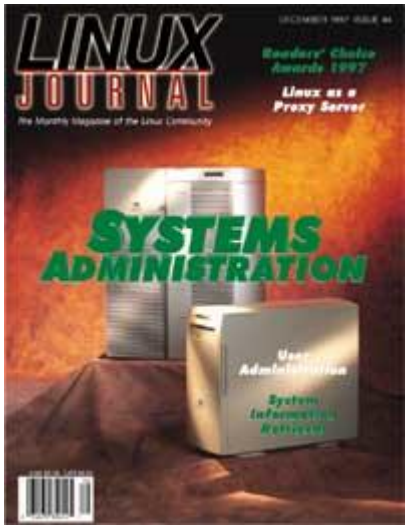
Linux Journal Issue #44/December 1997



Features

Pluggable Authentication Modules for Linux *by Andrew G. Morgan*
An implementation of a user-authentication API
User Administration *by David Bandel*
How to successfully manage your users.
Linux as a Proxy Server *by Peter Elton*
To protect your system, put your firewall on a proxy server.
Readers' Choice Awards 1997 *by Gena Shurtleff*
System Information Retrieval *by Dan Lasley*
Collect your system configuration files and store them on a separate machine.

News & Articles

Using Linux to Teach Unix System Administration *by Joe Kaplenk*
Linux Makes the Big Leagues, Hewlett Packard Interworks 97 *by Sam Williams*
The Quick Start Guide to the GIMP, Part 2 *by Michael J. Hammel*
LJ Interviews Larry Augustin *by Marjorie Richardson*

Reviews

Product Review OmniBasic *by Eric Harlow*
Product Review BRU 2000 for X11 *by Garrett Smith*
Product Review Raima Database Manager++, Velocis Database Server *by Nick Xidis*

Product Review Perforce Software Configuration Management System *by Tom Bjorkholm*
Product Review VA Research VAR Station II *by Jim Dennis*
Book Review STL for C++ Programmers *by Bob Adkins*
Book Review The Linux Multimedia Guide *by Michael J. Hammel*

WWWsmith

Industrializing Web Page Construction *by Pieter Hintjens*
Book Review CGI Developer's Resource *by Reuven Lerner*
At the Forge Keeping Programs Trim with CGI_Lite *by Reuven Lerner*

Columns

Letters to the Editor
From the Editor Promoting Linux
From the Publisher A Confession and Some Ramblings
Linux Means Business Linux in Camouflage
New Products
Kernel Korner The New Linux RAID Code *by Miguel de Icaza, Ingo Molnar, and Gadi Oxman*
Linux Gazette Disk Hog: Tracking System Disk Usage *by Ivan Griffin*
Best of Technical Support *by Gena Shurtleff*

Archive Index

Advanced search

LINUX JOURNAL

Pluggable Authentication Modules for Linux

Andrew G. Morgan

Issue #44, December 1997

This article describes an implementation of a user-authentication API: the Pluggable-Authentication-Modules API (PAM for short).

Have you ever wondered how much work it would be to modify login to work the way you want it to? Perhaps you want it to refer to shadowed passwords or you don't want to let users log in during certain times of the day. Perhaps your policy is to ensure that root cannot log in anywhere other than at the console; maybe, root should never log in when the system is connected to the network. You might implement MD5-secured passwords, or passwords secured with RIPEMD-160 or SHA? Perhaps you have decided passwords are insufficiently secure for your needs, and instead you would like a user to insert their identity card in a slot to log in. Further, you would like to integrate your system into a network security environment like Kerberos; the primary login of a user would need to activate their identity with respect to the Kerberos server. The variations and combinations are endless. There is no perfect solution. There never will be.

Given sufficient incentive to implement a new user-authentication scheme, the traditional solution has been to amend, or rewrite, all of the system-access applications (login, su, passwd, ftp, xdm, ...)--this takes time and resources. An annoying by-product of such an upgrade is that modifying one application requires you to simultaneously upgrade other applications to maintain a consistent entry policy for your system. Unless you write the applications yourself, it is not easy to simply insert the code and be sure you have filled all the holes.

The revision of a system's security policy also suffers from the following potential weakness: people who write appealing user interfaces are not always paranoid enough to write *secure* software. This point relates to both access-oriented (login, etc.) and information-security (e-mail encryption, etc.) software.

Current thinking on computer-related security is to separate the security policy from the service offered by an application. This allows the author of a security-dependant application to use an application programming interface (API) to take care of the security-related issues and concentrate their efforts on writing a good (robust but user friendly) application.

An API is typically defined by a document describing a set of functions that an application programmer can *rely* on. For example, **libc** is an implementation of a number of APIs. Collaborative organizations, such as ANSI, POSIX or X/Open, produce a document defining the API and then *vendors* (or enthusiasts in the case of Linux) implement it. Security-related APIs exist for tasks like user authentication and data encryption.

What is PAM?

Prompted by a comment on the linux-security e-mail list, Ted Ts'o asked the following question: “Has anyone thought of implementing PAM?”, and thus, he launched the Linux-PAM project. Marc Ewing (of Red Hat Software) quickly coded a framework for most of the library in January 1996. Since then I have been maintaining the Linux-PAM distribution. Thanks to the combined efforts of a number of people (I've included a “mostly” complete list at the end of this article), I'm proud to say Linux-PAM is now a reality. Better still, people are using it.

The document defining PAM is a request-for-comments paper (RFC) that was written by Vipin Samar and Roland J. Schemers III of SunSoft, Inc. Specifically, it is OSF-RFC 86.0, October 1995, “Unified Login with Pluggable Authentication Modules (PAM)”. The Linux-PAM URL at the end of this article contains a pointer to this document.

The PAM-API breaks the business of authentication into four independent management groups. These four groups are:

1. Authentication/credential acquisition
2. Account management
3. Session management
4. Authentication token (password) updating

Typically, a login application would need to use each of these groups when granting a candidate user access to a system. Applications like **passwd** only require access to the last of these groups.

The novelty and power of the PAM-API resides in the leading “P” for pluggability. It may surprise the reader to note that this is the part of PAM that

is irrelevant from the point of the login- application writer. Instead, it introduces a role for the system administrator in the process of choosing an authentication scheme for login.

Veteran Linux users will remember all the hype that surrounded the move to ELF from the older a.out system binary. A desirable feature of this transition was the introduction of a library function, **dlopen(3)**. This function call provides a reliable method for a running program to dynamically load some code for the purpose of execution. Its sibling function, **dlclose(3)**, is used to unload, or discard, such code when it is no longer required. Implementations of PAM utilize these functions to dynamically bind an application program to locally specified authentication modules. That is, the pluggability of PAM is dynamic and thus at the discretion of the local system administrator.

With a PAM-based login application, the system administrator can completely change the authentication scheme the application uses **without** modifying or recompiling it. In principle, this can even be done without rebooting the computer. However, before a new authentication scheme is fully deployed, the best approach is to isolate a computer from any network and test it under controlled conditions to ensure the new arrangement is robust.

Some details

In the remainder of this article, I will give a brief overview of how to write and use a PAM-based application. Potential authors and interested administrators should see the Linux-PAM URL (at the end) for more complete details. The intention of this article is to provide only a taste of what you can do with Linux-PAM. In particular, I will not address the issue of how to write an authentication module. For those details you should consult the full documentation available from the Linux-PAM URL.

Let us consider a generic login-type application. We will see which responsibilities are delegated to the PAM-API and which responsibilities are retained by the application. Finally, we will cover how a local administrator can configure the application to suit local taste.

Figure 1 is a graphic protraying the three components to a working PAM-based application. On the left is the application which is linked to the libpam.so shared library. In the middle we have the PAM library which parses the configuration file(s) and uses the entries listed there to load the configured selection of authentication modules (*PAMs*). Additionally, the application supplies a *conversation function* which provides a means for the modules to talk directly with the user.

In Listing 1 the skeleton of a login-type application is shown. It can be compared with the sample application given in the OSF-RFC defining PAM. The differences reflect enhancements to PAM since the RFC was written. Note, the listing is not very secure; it pays little attention to possible errors returned by the framework and is intended only to orient the reader.

The application initializes the library with a call to **pam_start()**, which silently parses the configuration file and loads those authentication modules that are appropriate to this application. It then enters a loop that attempts to authenticate applicant users. This process is repeated until a user is correctly authenticated, or the loaded authentication modules indicate that their patience has been exhausted.

Once a user has been authenticated, the **pam_acct_mgmt()** function is used to establish if the user is permitted to log in at this time. Modules of account-management type can be used to restrict users from logging in at certain times of the day/week or for enforcing password expiration. This latter case is intercepted, and the user is prevented from gaining access to the system until they have successfully updated their password with the **pam_chauthtok()** function.

The user's login session is surrounded with two sets of function calls. The outer function calls, **pam_open_session()** and **pam_close_session()**, mark the beginning and end of the PAM-authenticated session. Session initialization and termination typically include tasks such as making a system resource available (mounting the user's home directory) and establishing an audit trail. The inner function calls, **pam_setcred()**, first establish and finally release the PAM-configurable identity of the user. These can include credentials like access-tickets and supplementary group memberships.

Following logout, the user's PAM-configurable credentials are deleted, and the session is closed with a call to the pam_close_session() function.

Finally, with a call to **pam_end()**, the login application breaks its connection to the PAM library. The PAMs are unloaded, and the dynamically allocated memory is *scrubbed* and returned to the system.

This simple application demonstrates most of the functionality provided by the PAM paradigm. The conversation mechanism flexibly leaves the mode of direct interaction with the user entirely at the discretion of the application. In this way, it is possible for modules to be used simultaneously with graphically-based programs (**xdm** etc.) and their text based equivalents (**login** etc.).

Configuring an Application

Having obtained a PAM-based application, it is necessary to *attach* authentication modules to it. At the time of this writing there is an old way and a new way of doing this. The old way corresponds to the method advocated in the RFC and is based on a the contents of a single PAM configuration file: /etc/pam.conf. The *new* method is to break up the entries for the separate services into independent configuration files that are each located in the /etc/pam.d/ directory. The name of the file containing the configuration for a given application is the service name (in lower-case letters).

The function of the configuration file(s) is to provide a mapping from the application's service name to a selection of modules that provide authentication services to the raw application. In the case of the source program of Listing 1, the service name is simply “login”. (This is the first argument of the pam_start() function call.)

Along with similar entries for each of the PAM-aware services present on your system, the old configuration file (/etc/pam.conf) might contain entries of the following form:

```
...
# Here is the module configuration for login as it
# might appear in "/etc/pam.conf"
#
login   auth    requisite       pam_securetty.so
login   auth    required        pam_unix_auth.so
login   account required        pam_unix_acct.so
login   session optional        pam_cfs.so      \
        keys=/etc/security/cfs.keys
login   session required        pam_unix_sess.so
login   password        sufficient      pam_unix_passwd.so
login   password        required        pam_warn.so
#
...
```

The first four fields are: *service-name*, *module-type*, *control-flag* and *module-filename*. The fifth and greater fields are for optional arguments that are specific to the individual authentication modules.

The second field in the configuration file is the *module-type*, it indicates which of the four PAM management services the corresponding module will provide to the application. Our sample configuration file refers to all four groups:

- **auth**: identifies the PAMs that are invoked when the application calls pam_authenticate() and pam_setcred().
- **account**: maps to the pam_acct_mgmt() function.
- **session**: indicates the mapping for the pam_open_session() and pam_close_session() calls.
- **password**: group refers to the pam_chauthtok() function.

Generally, you only need to supply mappings for the functions that are needed by a specific application. For example, the standard password changing application, passwd, only requires a **password** group entry; any other entries are ignored.

The third field indicates what action is to be taken based on the success or failure of the corresponding module. Choices for tokens to fill this field are:

- **requisite**: Failure instantly returns control to the application indicating the nature of the first module failure.
- **required**: All these modules are required to succeed for **libpam** to return success to the application.
- **sufficient**: Given that all preceding modules have succeeded, the success of this module leads to an immediate and successful return to the application (failure of this module is ignored).
- **optional**: The success or failure of this module is generally not recorded.

The fourth field contains the name of the loadable module, pam_*.so. For the sake of readability, the full pathname of each module is not given. Before Linux-PAM-0.56 was released, there was no support for a default authentication-module directory. If you have an earlier version of Linux-PAM installed, you will have to specify the full path for each of the modules. Your distribution most likely placed these modules exclusively in one of the following directories: /lib/security/ or /usr/lib/security/.

The equivalent functionality for our login application can be obtained with the *new* configuration arrangement via an independent login configuration file:

```
#%PAM-1.0
#(The above "magic" header is optional)
# The modules for login as they might appear in
# "/etc/pam.d/login" this configuration is
# accepted by Linux-PAM-0.56 and higher.
#
auth    requisite       pam_securetty.so
auth    required        pam_unix_auth.so
account required        pam_unix_acct.so
session optional        pam_cfs.so      \
        keys=/etc/security/cfs.keys
session required        pam_unix_sess.so
password        sufficient      pam_unix_passwd.so
password        required        pam_warn.so
# end of file.
```

The newer configuration file is distinct from the old in that it is missing a *service-name* field. This field is not needed, as the name of the service-specific configuration file is by definition the *service-name* of the application.

It should be noted that the content of an /etc/pam.d/ directory takes precedence over the contents of any /etc/pam.conf file.

Note that the example contains more than a single module mapping for the **auth**, **session** and **password** management groups. This feature is referred to as *stacking* and enables a single application to make use of more than one module at a time. The order in which the modules are stacked is the same as the order in which they are invoked.

The two stacked **auth** modules are used to pam_authenticate() the user. The first module (pam_securetty.so) checks to see if the user is root and prevents root from logging in from an insecure terminal. The value **requisite** for *control-flag* is used to force immediate authentication failure if the **securetty** module fails. If this occurs, no more of the **auth** modules are executed. This has the benefit of preventing root from mistakenly typing a password over an insecure terminal line. Another popular module that can be used to prevent log in attempts like this is pam_listfile.so. It can be configured to perform many types of access control based on a list of tokens in a specified file.

When a non-superuser, **joe** for example, is successfully evaluated by the **securetty** module, control is passed to the next module in the stack, pam_unix_auth.so. This module performs standard Unix authentication. It prompts for a password and checks it against that stored in the local system. Providing your libc can handle it, it works on both shadowed and non-shadowed systems. An enhanced alternative to pam_unix_...so is the pam_pwdb.so module. This module makes use of the password database library, libpwdb.so, and can do things like MD5 passwords and offer RADIUS support. The important point is that the system administrator is the one deciding which authentication policy to implement by simply *plugging in* the corresponding module.

The **auth** module also supplies a binding for the pam_setcred() function. It is linked to the authentication process because the method by which a user is authenticated is strongly tied to the user's identity. Kerberos, for example, requires a network-based authentication and yields a *ticket* (the user's credential) with which they can obtain network services, such as remote login and print requests.

The **account** module line in our configuration file is used to check that the user is permitted to login. This is different from establishing whether the user is who they say they are. Account management deals with enforcing the expiration of passwords and preventing logins during system time. Our login example uses the standard pam_unix_acct.so module to enforce shadow password aging. Here, it is used (in conjunction with the **password** module type) to force the renewal of a user's password.

For experiments in this area, the administrator might like to try pam_time.so. This module can be configured to permit or deny access to users based on their terminal line, the time they are logging in and what they intend to do.

Next, we come to the session modules. The first in the stack, pam_cfs.so does not currently exist. (Because of ITAR export restrictions, I will not be writing it.) However, I have included it to illustrate the PAM session concept. At the start (and end) of the user's session on the system, this module would mount/unmount the user's cryptographically-secured home-directory, obtaining the user's home-directory/key-mapping information from the cfs.keys file. With PAM, someone could make a single module available, and that module could be used in any PAM aware application. The use of the value **optional** for *control-flag* ensures that the user can log in even when no such directory is available.

The second module, pam_unix_sess.so, logs a message with syslog(3) to announce the user's entry to and exit from the system.

Finally, we come to the **password** management group. Here, the stacked modules are invoked when users change their *authentication token(s)*. Traditionally, this change could be for updating their password, but it has the potential to be extended to refreshing a smart card or a yearly update of an employee's retinal scan. In the case of the login example, we simply request a replacement for the user's Unix password. Because the pam_unix_passwd.so module is marked as **sufficient**, a warning is logged by the pam_warn.so module only in the case that the user fails to successfully enter a new password.

In addition to configuring specific service names, there is also a default mapping, given the service name **other**. It can be used to ease the integration of new services by providing a default selection of modules appropriate to the local security policy. Instead, it can be used to deny access to any application that does not have a specific pam.conf entry. This is the recommended usage, for example, we can make use of the pam_deny.so (always deny access) and pam_warn (syslog(3) an informative warning) modules as follows:

```
#%PAM-1.0
#(The above "magic" header is optional)
# The modules for defaulting services as defined
# in "/etc/pam.d/other" this configuration is
# accepted by Linux-PAM-0.56 and higher.
#
auth    required        pam_deny.so
auth    required        pam_warn.so
account required        pam_deny.so
session required        pam_deny.so
password        required        pam_warn.so
password        required        pam_deny.so
# end of file.
```

This configuration always denies user access to an application. As before, the pam_warn.so is used to send a warning message to syslog(3) for administrative action. This configuration can be used to make sure that only specific services are available on your system. Note, if you write an application that uses PAM and this configuration file is not sufficient to block service from it, your application is not using PAM in the correct manner.

The Future

Thanks to the people at Red Hat, many applications are available that have been modified to support the PAM approach to user authentication. In this area, however, more work needs to be done. Indeed, some important applications are still without support. Most notably, the application **ssh** should be ported. Work is currently underway to provide a flexible X-based conversation function, and it is my feeling that this will lead immediately to a number of more friendly uses for PAM integration, such as popular games.

Most of the current Linux-PAM development is focused on the production of more powerful and varied modules. At last count more than twenty were written and the number is increasing—reflecting the variety of authentication schemes people use within the Linux community. You can be sure that with access to the corresponding hardware, someone out there will write an authentication module for a retinal scanner.

Recent work on the central PAM library, libpam.so, is directed at testing the security of our implementation. Due to a simple design that is well documented in the OSF-RFC and a friendly dialogue with those maintaining the Sun implementation, this is not proving to be an arduous task. In the six months it has been available, only one or two significant security problems have come to light. Currently, Linux-PAM is in beta-test (although the folks at Red Hat are doing an admirable job of offering production-level support for it). In particular, libpam.so.1.00 may even be available by the time you read this article.

Before getting to version 1.00, libpam will also have support for pluggable *password-mapping*, a method for chaining a number of different passwords together in a secure fashion. This concept (in a non-pluggable form) is discussed in the RFC, however, the X/Open group have since revised it, and we will implement their eventual specification in the coming months.

Further ahead, after releasing version 1.0, we will be modifying the syntax of the PAM configuration file. Current ideas relate to enhancing the *control-flag* field to be a great deal more flexible. Other changes are likely to be suggested and adopted as the number of administrators using PAM increases.

Finally, is Linux-PAM the only implementation of PAM? Is Linux alone? The answers to these questions are yes and no. At the time of this writing, the Linux implementation of PAM is the only fully functional version of PAM publicly available. This has been the case for at least half a year. However, a partial implementation of PAM is internally present (no /etc/pam.conf file) in Sun's Solaris 2.5. Rumors indicate that Solaris 2.6 will contain a complete, configurable implementation which should be available this year. The Sun implementation for PAM has been contributed to the X/Open group so look for it in other Unix variants in the coming years. If being cross-platform compliant is important to you, you should consult your vendor for more information. Alternatively, all of the source for Linux-PAM is freely available, so if your need is urgent, just read Linux as a synonym for *free* and try Linux-PAM on your other platforms now.

Acknowledgements

Resources

Update

Andrew Morgan is a postdoctoral researcher in Theoretical High Energy Physics at UCLA. His involvement with Linux-PAM has been in his spare time. The security of Linux-based systems is becoming his primary area of interest, and he is therefore pursuing a change of career. For Linux-related discussion, especially any security flaws in Linux-PAM, you can reach him at morgan@power.net.

Archive Index Issue Table of Contents

Advanced search

User Administration

David Bandel

Issue #44, December 1997

Mr. Bandel tells us all about users/groups, UIDs/GIDs, permissions and security. In short, how to successfully manage your users.

Once your system is installed and most of the services configured, it's time to add users. You may not have given this task much thought yet, but you should.

If you're like many Linux beginners, you may not understand the implications of giving someone access to your system. Perhaps, for some time, you've been using a Unix variant, in which the basic decisions have been handled for you. Indeed, some installation programs, such as Caldera's **LISA** (Linux Installation and System Administration), start the process for you by adding a user during the install process and then allow you to continue using it to add and delete users from the system. So what else is there to know?

For starters, adding users entails thinking about how you wish to administer the system overall. Security is directly affected by such things as groups and passwords. Their administration shouldn't be left to chance or the whims of the installation program—at least, not without exploring their impact.

/etc/passwd

When you add a user, however you do it (I'll mention some programs later), you add an entry to the /etc/passwd file. This file consists of seven colon-separated fields, specified all on one line:

```
col:JkH7XmXwH6e/E:100:100:Bandel:/home/col:/bin/bash
```

The fields are: user login name, password, UID, GID, comment (gecos) field, user's home directory and user's default shell.

If you decide to edit the /etc/passwd file manually, please be sure to make a backup copy first, so that if something goes wrong, you can start over. In

general, I recommend using one of the programs specifically designed to edit the /etc/passwd file.

1. Login Name

The first field is fairly self-explanatory. The system matches this entry with the next field as the user's login name/password pair. Each name must be unique. This is the only field required to be unique; I will explain the implications of keeping certain other fields unique below. Beyond that, making the user name meaningful helps to quickly identify who sent a 10MB file to the printer when the boss needs a hard copy of his budget for a Board of Directors meeting in ten minutes. In most Unix variants, including Linux, the login name is limited to 8 characters.

2. Password

The second field is the password. This field can contain any of a number of things, but normally contains an encrypted password. The encrypted password must be 13 characters in length, composed of a two character seed plus the encrypted password. Other entries often seen in the password field include: “*”,“LOCKED” and “VOID”. If you see an “x” in this field, you'll want to read about shadow files below. These, or any other incorrect entries, effectively lock this user out. If you wish to lock a user out without losing the password they are using, you can insert an apostrophe as the first character. An apostrophe can never be used as part of the “scrambled eggs”, so apart from being illegal as part of a seed, you have also made the field size 14 characters. To allow the user access, just remove the apostrophe.

Also note that this field might be empty. Always check the /etc/passwd file to ensure that the second field is not empty. A field is defined as empty or NULL if the two colons delineating this field are next to each other with nothing in between, not even a space. If it is empty, no password is required to log in. The implications of this are obvious.

3. UID

The third field is the UID (user identification). Valid UID ranges are from 0 to 65534. A UID of 0 has a special significance to the system; it is the omnipotent “superuser”, a term derived from the **su** (substitute user) program normally used by a normal user to become root. When invoked without an argument (user name) or with only a hyphen (**-**), root is the default.

A UID of 65534 is commonly reserved for “nobody”, a user with no privileges as opposed to a non-privileged user. This UID is often used for individuals accessing your system via FTP or HTTP.

Custom dictates that the numbers 1 to 99 are reserved for system users, such as wheel, daemon, lp, operator, news, mail, etc. These numbers are more than sufficient for most system requirements. Also, you may want to reserve 100 to 999 for semi-privileged (for lack of a better term) users. These users are administrators who don't need total root powers, but who do some administration and thus need more privileges than those given to non-privileged users. Reserve 1000 to 9999 for local users and 10000 to 65534 for remote users (if any).

Reserving groups of numbers for particular users helps if you ever need to search through logs for suspicious user activity. Some distributions, like Caldera, start UIDs at 100, others, like Red Hat, at 500, and still others, like Debian, at 1000 for non-privileged users. Just decide on a scheme and stick to it. Since each distribution does its own thing, in order to mix distributions within an organization, you have to intervene manually if the systems are networked. This is particularly true if you want to use NIS or NFS to mount common directories (e.g., /var/spool/mail, /home).

Contrary to popular belief, the UID field does not have to be unique. In fact, one method that keeps “wanna-be” system crackers busy is to use the technique discussed above and put an apostrophe as the first character in the password field for root. Then create another entry, perhaps tuber, with a UID of 0. Some systems require the root entry to exist and be the first entry.

But beware: a non-unique field can cause problems. For example, if you are logged in as tuber (UID 0) and activate a password-protected screen lock, when you attempt to unlock it, you will find you've outsmarted yourself. The kernel knows you by your UID; it looks up UID 0 beginning with the top of the /etc/passwd file and finds it as the first entry corresponding to “root”--that's where it stops; it doesn't look any further. But root's password has been locked with an invalid entry. At this point, your choices are to telnet in and kill the screen-lock process or to reboot the machine. The other possibility for getting out is to use an unlocked virtual terminal if you have one running to which you can switch. (Sometimes, the “extra” virtual terminals come in handy.)

Before I leave UIDs, here are two more things to think about. First, if you have any “rhosts” files with references to other systems outside your own network, make sure your user names are unique not only in your system, but between systems. If you have a jonesd (for Donna Jones) and the other system has a jonesd (for Danny Jones), and they use **rlogin** to remotely log in to each other's systems, they'll have access to each other's files (as the other person). These files can cause many problems and probably should not be used.

The second thing is the **find** command, which is used extensively by systems-administration programs and can be used to find user's files for deletion or changing owners (**chown**) before (or after) the files become orphaned. Learn this command, and incorporate it into your tool box. See the side bar “A Few Words About the Find Command”.

A Few Words about the Find Command

4. GID

The fourth field is the primary GID (group identification) number. Every user belongs to a group. Some distributions use a common group such as “users” to which everyone belongs. Others create “private groups” and assign each user to his own private group as their primary or default group. Caldera uses a common group, while Red Hat and Debian use private groups. I would recommend reading the Red Hat Users Manual, available via anonymous FTP from ftp.redhat.com/pub/redhat/redhat-4.1/Users-Guide/ in various formats. A discussion of private groups as they relate to easing the system administration burden can be found in this manual.

To discover the groups they belong to, users can type **groups** at the command prompt. The first group name returned is the default group. Files saved by that user will, by default, have the group identifier set to the first value returned unless the subdirectory has the “setgid” bit turned on.

5. Comment

The fifth field is a comment, which usually contains the user's full name. You may see this field referred to as a gecos (General Electric Comprehensive Operating System) field. Shadow files sometimes use this field for something other than a user's name, but only under special circumstances.

6. Home Directory

The sixth field is the user's home directory. The **cd** command given with no argument or with the tilde (~) as an argument will move the user to this directory. This directory must have a full path name, i.e., start at the root and go down. One notable exception to this rule is the system user “bin”, which has a relative path of bin as the home directory, because several bin directories exist on the system.

7. Default Shell

The last field is the user's default shell. Most Linux distributions (in fact, all of those I am aware of) default to bash, the Bourne Again Shell. For most users,

this is probably the best overall shell. C programmers tend to like the syntax of one of the C shells (csh or tcsh), but for writing scripts, most users will do better with bash. If you need a user to have a system entry for any purpose (perhaps a WWW database account), but don't want him to be able to log in, you can specify /bin/false as his shell (as is usually done with the nobody account). The shell can also be a program that runs when the user logs in, but it would be the only thing the user could do—exiting the program would log him out. If you choose this option, be aware that some programs allow users to start a shell from within the program. If the program "shells out" to a true shell, then the user has a back door into the system.

Shadow Files

A complete discussion of shadow files is beyond the scope of this article. However, just to give you some idea of what they are and why they were created, I offer the following. The /etc/passwd file is "world" readable; anyone can make a copy of it. "So what," you say, "the passwords are encrypted and can't be used." Not true. Most users, in particular those who choose their own passwords, choose passwords that can be guessed or broken by a "dictionary attack" --that is, by using a program such as **cops**, an attacker can run a dictionary through **crypt** using the seed from the password to see if they get a match.

When the shadow file is used to maintain passwords, it is not world readable and cannot be copied off the system. An "x" in the password field tells you that this system uses shadow passwords. The shadow file also contains information about when the password was last changed, if the user will be forced to change it and how often (password expiration date), etc. Periodic forced password changes reduce risks posed by users who do not guard their passwords well.

Most recent Linux distributions include the PAM (pluggable authentication modules) libraries. PAM doesn't require applications to be "PAM aware" in order to function correctly, but these are only libraries and do require programs written with them. The shadow suite requires numerous programs to be recompiled or replaced with shadow-aware applications. If you need this kind of security, you need to do some serious reading.

/etc/group

The /etc/group file is also a colon-separated list consisting of only four fields: group name, group password, GID and members. The group name/password pair works the same as the /etc/passwd file. However, groups usually don't have passwords associated with them. A quick look at your /etc/group file will reveal the second field doesn't have an entry. Adding passwords to groups

doesn't normally enhance security, since users are even more lax about group passwords (why not, everyone knows it) than their own.

The original group implementation in Unix allowed a user to belong to only one group at a time, and they changed group if required. Now, belonging to several groups at a time is common, so this has little relevance in most situations. Files are saved as the users' primary group, but can be changed manually using the **chgrp** command or via the setgid bit on the directory. The third field is the group-identification number and is functionally equivalent to its /etc/passwd UID counterpart. The fourth field consists of a comma separated list of group members (no spaces are used following the comma).

One last note on the /etc/passwd and /etc/group files. Several distributions I've worked with lately have been adding a line to the bottom of the file like this: +:::::: (The /etc/group file will only have four colons.) This line is added so that NIS databases are appended if no matching entries are found for user/password pair. This **must** be the last line in the file. Entries following this line are not read. I highly recommend that if you are not using NIS, you delete this line and use **chmod** to change permissions on **ypbind** so that it is not executable. If you run ypbind without a properly set up NIS server and the proper databases, you are vulnerable to someone who can trick your machine into binding to his server and reading his NIS maps to gain unauthorized access to your system.

Adding and deleting users/groups

All of the Linux distributions come with facilities for automating the process of adding, modifying and removing users. All include shell scripts or programs such as **adduser**, **useradd**, **usermod** and **userdel**. Some distributions, notably Caldera and Red Hat, come with additional programs to do the job. Caldera uses its **LISA** program to handle adding and deleting users. Red Hat's **usercfg**, for use in a graphical (X11) environment, not only handles user additions and deletions, but also helps manage groups.

All accomplish certain basic chores beyond just adding a line to the /etc/passwd file and appropriate entries to the /etc/group file. Given the sensitivity of the /etc/passwd file and the consequences of corrupting this file, that alone would warrant its use. A system no one (not even root) can log into isn't much use. In fact, if you do edit /etc/passwd with an editor, you shouldn't keep it open long. This file is accessed by numerous programs for login and authentication purposes, and holding this file open (especially if another user decides to change his password) could cause system instability.

All these scripts and programs prompt you through the information needed to create the new account. If you need to add or delete group accounts, the

groupadd and **groupdel** (**delgroup**, **addgroup**, **groupmod**, etc.) programs can be found on most systems to accomplish this task.

The use of Network Information Services (NIS), formerly known as the Yellow Pages (a trademark of British Telecom) and still carrying the “yp” prefix on the programs, will necessarily change some of the procedures I've discussed, but the basics remain the same. NIS is used at larger sites to ease the administrative burden of pushing changes to each machine. But accounts still need to be set up. NIS just centralizes and distributes files from the master server instead of changing files on each machine individually. If you do work at a large site that uses NIS, you'll still have some normal accounts set up on each machine for those times when maintenance requirements dictate bringing the machine up in single user or non-networked modes.

Security

Since I've been mentioning security from the beginning of this article, I thought I would mention a few programs to help you maintain security on your system.

The first program, **npasswd** (no relation to the file created by reversing the shadow password conversion), does some sanity checking when users change their password. This program is used in place of the regular **passwd** program by renaming the passwd binary (/usr/bin/passwd not /etc/passwd) to something like passwd.orig and creating a link to npasswd from passwd.

Another program you may be interested in is **pwgen**. This program generates random passwords containing mixed upper and lower case characters, numbers and legal (for a password) punctuation. Personally, I find these passwords too difficult to remember, and since writing them down is a security no-no, I devise my own (hopefully) secure passwords.

I mention the **pwdutils** package only for those using NIS. This package contains NIS compatible programs. Those contemplating using NIS or shadow files need to ensure that the programs they use for system administration (particularly user administration) work with NIS or the shadow suite. Above all, before you begin installing either NIS or the shadow suite, RTFM—read the fine manual (and keep an emergency boot disk handy).

/etc/skel

Among the myriad things you can do that will make your life easier as a systems administrator is to maintain certain files and subdirectories. One such subdirectory that is often overlooked is /etc/skel. This subdirectory is the skeleton used by all (that I'm aware of) scripts and programs to create the new user's $HOME directory. You've seen how administration programs add users

to /etc/passwd. You've noticed that at the same time a /home/$USER subdirectory is created with proper permissions and a few common files, such as .bashrc, .cshrc, etc. All this is done by copying /etc/skel with all its files and subdirectories to the user's new home directory and changing the ownership (chown) to the new user. Use the /etc/skel directory to keep all the files you wish new users to have—maybe a default .fvwmrc for their X sessions or some changes to their .bashrc files. Then the new user will be ready to start working as soon as you've run the **adduser** program.

If you view some of the programs I've discussed, you'll note some are binaries and some are scripts. I prefer to use a modified adduser script in which I have included a couple of lines for something I occasionally forget. My Caldera boxes run a modified **fvwm** (from LST) that writes to the file /var/run/xlaunch/$USER/background-:0.0. While not a fatal error if the directory doesn't exist, I don't like to see error messages scrolling through my console box. So, my modified script includes two lines to create this subdirectory and then **touch** the file name. You can modify your scripts in a similar manner.

Conclusion

Once set up, user administration is not that difficult. The necessary tools are installed by default, and some nice GUI programs make the task almost enjoyable. A well-thought out user-administration policy will help you enhance security while making the system friendlier and easier to use for both you and the users. Create some accounts and introduce your friends to the joys of “Linuxing”.

David Bandel is a Computer Network Consultant specializing in Linux, but he begrudgingly works with Windows and those “real” Unix boxes like DEC 5000s and Suns. When he's not working, he can be found hacking his own system or enjoying the view of Seattle from 2,500 feet up in an airplane. He welcomes your comments, criticisms, witticisms and will be happy to further obfuscate the issue. He can be reached via e-mail at dbandel@ix.netcom.com.

Archive Index Issue Table of Contents

Advanced search

Linux as a Proxy Server

Peter Elton

Issue #44, December 1997

How do you keep unwanted visitors out of your network while still giving your users the Internet access they rely on? The answer is a firewall equipped with a proxy server.

Proxy servers are software applications that run on your firewall machine in order to provide indirect Internet access to your network. The firewall can be either a “single-homed” host or a “dual-homed” host. A single-homed host is a machine with one network card. This configuration relies on the Internet router to block all packets to any machine except the firewall. A dual-homed host is a machine with two network cards that has routing capabilities disabled. Computers behind the firewall can talk to the dual-homed host, and computers on the Internet can communicate with the dual-homed host. However, since routing between the network cards is disabled, the computers behind the firewall cannot talk directly to the computers on the Internet.

The proxy server is used to allow Internet access from inside the protected network through either the single or dual-homed host firewall. The client applications speak directly to the proxy server and the proxy server in turn speaks directly to the Internet hosts on behalf of the client, thusacting as a proxy. This interaction allows Internet access to all clients on the internal network, but leaves only one machine, the firewall, directly vulnerable to attacks from the Internet.

The proxy server takes a packet from inside your network that is bound for the Internet and changes the “from” address to its own address. It then forwards the packet to the destination host. The beauty of the proxy server is that the destination host thinks it is talking only to the firewall. When the firewall receives the response from the destination host, the proxy server sends the packet back to the original requesting machine. The client has the illusion that it has been communicating directly with the host on the Internet. The host on the Internet has the illusion that it is only dealing with the firewall.

This method is a big advantage when you access FTP sites that do double-reverse lookups. These sites, as a security measure, want to ensure you are truly coming from the address you've given. The host name of the requesting IP address is looked up in the DNS records. The server then does a lookup of the host name it received. If the IP address it receives from this last lookup does not match the requesting one or if the DNS lookup failed to find any entries, the server denies access.

If you are denied access to one of these sites, there is most likely a problem with your DNS setup. When you have to manage several machines across your network, keeping all the entries up to date can be a daunting task. With a proxy server in place, your entire network appears to come from the IP address of the proxy server, thus reducing the total number of properly configured DNS entries you need.

Another advantage of using a proxy server is that since all outbound traffic must pass through the firewall, as an administrator, you can monitor which types of Internet activity are occurring. The proxy server has very robust logging capabilities which allow you to see who is accessing what on the Internet. Attempted access from the outside is also logged closely.

Setting up the Linux box

I will not go into the details of setting up a packet-filtering router, since that type of information is vendor specific. However, I will give you the basic information on setting up a dual-homed host firewall. Assuming you use a Linux machine for your host, you will need to have two network cards installed in your machine. Read the “Multiple-Ethernet” mini-HOWTO located at ftp://sunsite.unc.edu/. I used two 3Com509 cards.

Auto-sensing the modules to load is a common problem when using two identical cards, so I compiled the drivers into a monolithic kernel instead of a modular one. I also added the following line to my /etc/lilo.conf file:

```
append="ether=11,0x300,eth0 ether=10,0x270,eth1"
```

This ensures that the proper parameters are passed at boot time.

Configure your kernel to keep it from routing IP packets (see Listing 1). To further ensure protection and anonymity, use one of the “bogus” class addresses (see Table 1) as per RFC1918. These IP addresses are set aside by the INTERNIC for use behind a firewall. Any packet with one of these IP addresses is dropped by the Internet backbone routers. See Figure 1 for an example of a network topology with a dual-homed host firewall. The example configuration files in this article are based on this basic topology. Our protected network is

assigned the “bogus” Class C address 192.168.50, and we assume that the valid IP address of the Internet side of the firewall is 111.222.333.1.

Listing 1

Table 1

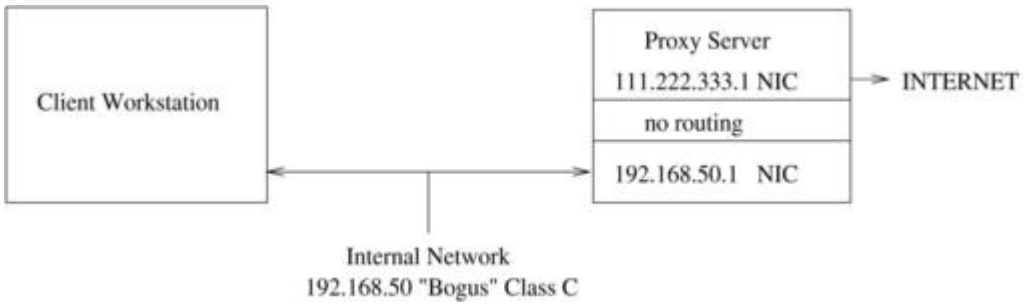


Figure 1. Dual-homed Host Firewall

The Socks5 Proxy Server

The Socks5 server is freely available from http://www.socks.nec.com/. There are several advantages to using the Socks5 server. Many TCP/IP applications have support for Socks5 proxies built in. There is an INTERNIC RFC for it (RFC1928). It proxies all services through one port, allowing you to block incoming packets on most other ports. Finally, it has support for the most commonly used services that your users will want: HTTP, FTP, TELNET, finger, archie, whois, ping and traceroute. Unix clients are included and compiled with the source distribution. A client application for Win 3.11, Win95 and WinNT is also available for download. If you have custom applications, you can use the Socks5 library to compile Socks5 support into your application.

Compiling

I was able to compile the source distribution for Socks5 correctly the first time. A configure script is used to set up all the necessary flags, parameters and Makefiles for your system. Afterward, it's as simple as executing **make** and then **make install** to put all the binaries and man pages into the /usr/local/ directory tree. The following are the steps required to build and install the Socks5 software:

```
tar -xvzf socks5-beta-0.17.2-exportable.tar.gz
cd socks5-beta-0.17.2-exportable
./configure
make
su
make install
```

Configuration

The server can be started via **inetd** or run as a daemon. Running as a daemon has the advantage of increased performance to the user. Running via inetd leaves the firewall less burdened when not in use. If your site is like mine, there is never a time when the Internet is not being accessed. I configured the Socks5 server to run as a daemon and added the command to start the server to my /etc/rc.d/rc.local file.

Configuration of the firewall is done in two steps. First, there is a configuration file on the server that must be set up specifically for your site. The default file is /etc/socks5.conf (see Listing 2). The man page gives information on the appropriate syntax, and there are also example configurations at http://www.socks.nec.com/v5examples.html. Second, there are configurations that must be done on each of your client workstations. On Unix clients, this is the /etc/libsocks5.conf file (see Listing 3).

Listing 2

Listing 3

On your Win-based machines, several different things need to be done. If all of your users limit their Internet usage to the Web, you can keep your configuration limited to the options available in both Netscape Navigator and Microsoft Internet Explorer. For Netscape Navigator, the appropriate settings are located in "Options"**->** "Network Settings"**->**"Proxies". Select "Manual configuration" and then enter the Socks5 server IP address with port 1080 (note: this is the default port, and can be configured differently at compile time). For Netscape Navigator 4.0, the settings are found under "Edit"**->** "Preferences"**->**"Advanced"**->** "Proxies". The remainder of the configuration is the same as above. For MS Internet Explorer, select "View"**->**"Options"**->** "Connection". Select "Connect through a proxy server". Enter the IP address of your Socks5 server as well as the port 1080.

If your user's demands go beyond simple Web access, the download site for the Socks5 software also contains two versions of SocksCap, the Windows redirector: SocksCap16 and SocksCap32. The SocksCap16 software is used for Windows 3.11 clients while SocksCap32 is used on both Win95 and WinNT. The SocksCap16 application only needs to be running at the same time as the Winsock application in order to proxy the application. The SocksCap32 application, however, must be started first, and the Winsock application launched from within SocksCap32. Alternately, you can create a shortcut to the desktop or the "Start" menu that calls the Winsock application profile from the command line:

```
C:\Program Files\SocksCap\sc32.exe ws_ftp
```

Both versions of SocksCap require you to enter the appropriate IP address and port to your server when you start the application for the first time.

The TIS Firewall Toolkit

The Trusted Information Systems Firewall Toolkit (TIS fwtk) is another widely-used, freely-available, proxy-server solution. The TIS firewall toolkit provides very specific proxies for each service, giving you the ability to set up just an HTTP proxy server, for example, if you wish to limit your users to just that service. When the package builds, the proxies that are built include an HTTP (http-gw), FTP (ftp-gw), TELNET (tn-gw), rlogin (rlogin-gw), X (x-gw) and generic proxy (plug-gw). Also included is a secure replacement for sendmail (smap) as well as an authentication module (authsrv). The generic proxy gives you the ability to configure proxies for specific machines and ports. Possible uses for this proxy could be proxying Usenet news as well as accessing e-mail through the POP3 protocol. (Socks5 does not include support for either News or POP3.)

Compiling

The TIS fwtk builds fairly easily. I had to apply the http-gw patch in order to get the HTTP proxy to build. If you are building this toolkit on a system other than Linux, make sure you use **gmake** instead of **make**. I ran into this problem when I tried to build this package on an SGI. Doing this may require you to first acquire and build GNU **make**, which is available at ftp://prep.ai.mit.edu/pub/gnu/.

There is no configure script available with the TIS fwtk. Instead, there are several versions of the Makefile.config. Simply apply the http-gw patch, move Makefile.config.linux to Makefile.config, run **make** and then run **make install**. Note that in order to build the x-gw X proxy, you need the Motif libraries. The easiest way to get them is to download **lesstif**, a Motif clone available at http://www.hungry.com/products/. The following are the steps required to build and install the fwtk software:

```
tar xvzf fwtk-2.0.tar.gz
cd fwtk
chmod -R 755 *
tar xvf ../http-gw.patch.tar
mv Makefile.config Makefile.config.orig
ln -s Makefile.config.linux Makefile.config
makea
su
make install
```

TIS fwtk Configuration

The binaries are placed in the /usr/local/etc directory, a location not likely to be affected by system upgrades. The proxies can be configured to run as daemons or by inetd. Take careful note of the man pages. There are specific command-line arguments that must be invoked in order to get the proxy to run as a daemon. As I mentioned above, you can configure as many or as few of the proxies provided. The configuration is much more complicated than for the Socks5 server. The configuration file is called /usr/local/etc/netparam and is parsed based on the proxies. Each line starts with the name of the proxy followed by a colon and then by the options (see Listing 4).

Listing 4

The client configuration is different from the Socks5 configuration. For the two browsers mentioned above, simply leave the Socks5 entry blank and fill in the HTTP proxy and port, as well as FTP and TELNET entries, if you have set these services up on the firewall. Good news for those using a browser other than Navigator and Explorer—you can use the TIS fwtk http-gw proxy with any browser. Simply prefix all web addresses with http://*IP-or-name-of-firewall-host/*. For FTP, you first FTP to the proxy-server host. When prompted for the user name, enter anonymous@ftp.ftpsite.com; the proxy then goes out and makes the appropriate connection. For News, you must configure the plug-gw as illustrated in the configuration file (see Listing 4). Accessing news is as simple as configuring your news client to point to the proxy server instead of the real news server.

Integrating both packages

When I first set out to construct a firewall/proxy server solution where I work, I initially chose the Socks5 server. (I had downloaded them both, but the Socks5 server compiled without errors, thus sealing my choice.) Within a day, I had figured out which settings I needed for my network and had the server running as a daemon.

Getting users excited about using the proxy server as opposed to direct Internet access is not easy. Recently, a user approached me ranting about how he could not stream video in from the Internet. After he calmed down, I asked him which URL he was trying to access. Then I quickly found a FAQ for the particular plug-in he was using with an entry similar to “How do I stream video through a firewall.” After reading the FAQ and looking at the available options, I found that this particular plug-in did not have Socks5 support. It did, however, have support for general HTTP proxying. So, I turned my attention back to the TIS fwtk I had abandoned 10 months before. I found a newer version, applied

the http-gw patch, executed **gmake** and had the toolkit running later that day. The video streaming worked beautifully.

This story illustrates the fact that you do not have to make an either/or decision about Socks5 or the TIS firewall toolkit. The two packages can be used together to give your users a wide variety of Internet access.

Conclusion

Setting up a proxy server is a great way to give your users Internet access while still protecting your network from Internet attacks. Between both the Socks5 server and the TIS firewall toolkit, you can give your users as much or as little Internet access as you want.

Resources

Peter Elton has a BS degree in Computer Science from UNLV. When his pupils are not glazed over from viewing a monitor too long, he enjoys sailing, offroading and spending time with his wife and two kids. He can be reached at elton@rridge.com.

Archive Index Issue Table of Contents

Advanced search

1997 Readers' Choice Awards

Gena Shurtleff

Issue #44, December 1997

Linux Journal readers pick their favorite Linux software, hardware and products.

Welcome to *Linux Journal's* third annual Readers' Choice Awards.

In response to requests for a larger survey, last year's nine categories were expanded to twenty. A few of the new categories are “Favorite Shell”, “Most Desired Upgrade” and “Best Linux Web Page”.

The Readers' Choice survey was conducted on the *Linux Journal* web site, where the voting was open for 6 weeks. Almost 3000 people participated in this year's poll. This survey, though unscientific, is a good way to see what products *Linux Journal* readers are using, what books they are reading and what games they are playing.

Now, here are the results...

Favorite CD-ROM Distribution

Winner: **Red Hat Linux**

Runner Up: Debian Linux

Red Hat was the runaway winner of the “Favorite CD-ROM Distribution” this year. Red Hat garnered almost three times as many votes as its nearest competitor, Debian.

Favorite *Linux Journal* Column

Winner: **Kernel Korner**

Runner Up: Best of Technical Support

Kernel Korner, our technical guide to kernel programming, won the prize as *LJ* Readers' favorite column. Second place went to the question and answer column Best of Technical Support.

Most Used Development Tool

Winner: **GCC**

Runner Up: Perl

Not surprisingly, GCC won as the “Most Used Development Tool”. Perl came in a strong second.

Primary Communications Board

Winner: **Cyclades**

Runner Up: Digi International

For the third year in a row Cyclades' boards have topped our “Primary Communication Board” category.

Favorite Shell

Winner: **Bash**

Runner Up: Tcsh

Bash was the clear winner in the competition of the shells. Our readers picked Bash almost four times more often than any other shell.

Most Used Text or Word Processor

Winner: **vi**

Runner Up: Emacs

vi edged out Emacs in this hotly contested category. The word processing programs lost big to traditional text editors—most word processors got less than a tenth of the votes that vi received.

Favorite Graphics Program

Winner: **GIMP**

Runner Up: Corel Draw!

GIMP, the freely distributed graphics program, won a decisive victory over the other graphics applications. GIMP had an edge of more than 1000 votes over runner up Corel Draw!

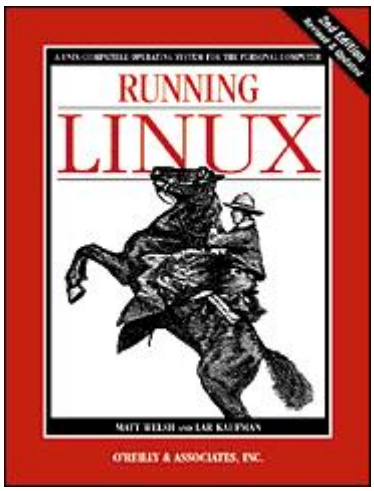
Favorite Web Browser

Winner: **Netscape**

Runner Up: Lynx

Linux Journal's Readers favor Netscape as their web browser by a wide margin over the text-only Lynx browser.

Most Indispensable Linux Book



Winner: *Running Linux* by Matt Welsh and Lar Kaufman \tPublished by O'Reilly & Associates, Inc.

Runner Up: *Linux Network Administrators Guide* by Olaf Kirch \tPublished by SSC, Inc and O'Reilly..

In a reversal of last year's results, *Running Linux* took first prize and the *Linux Network Administrator's Guide* took second.

Best Linux Web Page

Winner: **Linux v2 Information Headquarters** \thttp://www.linuxhq.com/

Runner Up: The Linux Documentation Project \thttp://sunsite.unc.edu/LDP/ \thttp://www.ssc.com/linux/resources/ldp.html

With so many excellent Linux web pages out there this category was quite competitive. The Linux v2 Information Headquarters won by just a few votes with the Linux Documentation Project close behind.

Most Desired Upgrade

Winner: **ISDN**

Runner Up: Laser Printer

The Most Desired Upgrade was for faster connectivity through ISDN. Second on our Readers' wish list was a laser printer.

Most Played Linux Game

Winner: **Quake**

Runner Up: Doom

Quake, the 3-D monster killing fest, topped our Readers' list as their favorite Linux Game, followed by Doom, another 3-D monster killing fest.

Primary Platform

Winner: **Intel**

Runner Up: PowerMac

The Primary Platform that *LJ* Readers are using is, not surprisingly, Intel. Making a good showing in second place was Linux on the PowerMac (MkLinux).

Favorite Mailer

Winner: **Pine**

Runner Up: Elm

This category came down to a competition between the two most common mailers, Pine and Elm. Pine was the decisive winner, receiving over twice as many votes as Elm.

Primary X-Server

Winner: **XFree**

Runner Up: Accelerated X

XFree is the X-server of choice amongst voters in this poll. The free X-Server XFree topped second place Accelerated X by over 2000 votes.

Primary Window Manager

Winner: **fvwm**

Runner Up: fvwm-95

The clear winner in the Window Manager category is **fvwm** with the Windows95 look-alike **fvwm-95** in second place.

Best Database

Winner: **SOLID Server**

Runner Up: Empress RDBMs

SOLID Server was a clear winner in the very competitive database category garnering almost three times as many votes as its nearest competitor, Empress RDBMs.

Favorite File Manager

Winner: **xfm**

Runner Up: Northern Commander

xfm won as Favorite File Manager. Northern Commander finished second.

Best Backup Utility

Winner: **BRU**

Runner Up: CTAR

LJ Readers' voting shows that they prefer to back up their data with BRU.

Favorite Audio Tool

Winner: **RealAudio**

Runner Up: TiMidity

From a list of over 20 possibilities Real Audio (first) and TiMidity (second) were the winners in this category.

More information about the products and programs included in this article (or other Linux hardware and software) is available through our Linux Resources web site at http://www.ssc.com/linux/.

Archive Index Issue Table of Contents

Advanced search

System Information Retrieval

Dan Lasley

Issue #44, December 1997

Mr. Lasley gives us a handy script tha he created to collect system configuration files for one machine and store them on another.

In issue 39 of *Linux Journal* (“Is Linux Reliable Enough?”, July, 1997), Phil Hughes writes about down time due to the failure of a hard disk:

> At some point we had a configuration disk for our firewall; but when we needed to replace the hard disk, the configuration disk had vanished. This loss cost hours of work time and probably a day of uptime. Having a complete backup of everything, boot disks for all machines, spare cables and disk drives and other assorted parts can make a big difference in the elapsed time to deal with a problem.

I've developed a script to simplify the kinds of Linux system administration difficulties which Mr. Hughes describes. I use the script on all my Linux systems and feel it would benefit other system administrators as well as Mr. Hughes.

The Problem

I've installed Linux on four Intel Pentium-based systems and seven Intel 486-based systems. All of the 486-based systems had previously been abandoned because they had neither sufficient processing power nor sufficient memory for Windows for Workgroups, Windows 95 or Windows NT, my company's choices for a desktop operating system. All of these 486-based systems run Linux very capably.

I use these Linux systems for network troubleshooting, testing, research, evaluation, experimentation and program development. Installing and using Linux in a large corporate enterprise has helped me learn more about DNS, networking, network programming, HTML and HTTP, system administration and other aspects of the Unix environment.

Although these Linux systems have been extremely useful, the age and diversity of the equipment involved makes system-administration tasks difficult at times. Consider the mix of equipment shown in Table 1, “Linux Systems and Major Components”. (This table also provides a list of the names of the Linux systems I'll be referring to throughout this article). The permutations of five computer vendors, three disk types, seven types of networking cards (the five NE2000 clones are from three vendors), and four CD-ROM types create some interesting installation, configuration and administrative headaches.

I've encountered other significant, system-administration difficulties as well:

- The various hardware components of these systems change from time to time as research and evaluation needs dictate.
- Because I am trying to win acceptance of Linux within my organization, I perform most of the system-administration functions on my own time.
- None of these systems have a working tape backup unit.
- These systems are distributed among three locations within the Memphis area. All are interconnected via a metropolitan area network that forms the basis for a method of simplifying system-administration duties.

As if these issues weren't serious enough, soon after installing my sixth Linux system, its hard disk began failing. Since the disk was failing slowly, I had time to recover all the pertinent configuration information to enable me to reinstall and reconfigure Linux quickly after I replaced the failing disk.

The Solution

Listing 1 shows a shell script I created to ease the chores of maintaining multiple, disparate Linux systems. The script, which I call **collect**, uses remote shell commands (**rsh**) and remote copy commands (**rcp**) to copy a number of files (which are described briefly in the “Collected Files” box) from a remote Linux system to “cuthroat”, my primary system-administration system.

Collected Files

If I lose any Linux file system (except for cuthroat's), I don't have to be concerned about losing important configuration information. As we'll see later, since I propagate all the collected information on cuthroat to several other systems, I don't have to worry about losing cuthroat's file system.

Using the Collect Script

After writing and testing the collect script, I created the /admin directory on cuthroat and moved the script to this directory. When I wish to collect system-administration information from a Linux system (barb, for example) and store

that information on cuthroat, I log on to cuthroat and type the following commands:

```
cd /admin
collect barb
```

If the /admin/barb directory doesn't exist, the collect script creates it, and then begins copying the remote system's files. In the spirit of UNIX brevity, the only screen output is a single line:

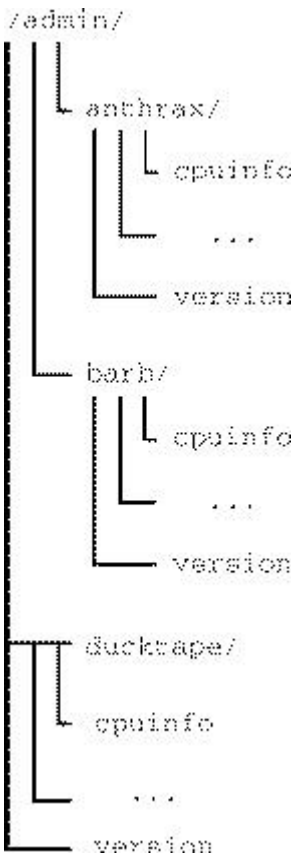
```
barb:  copying /proc, .config, lilo.conf, partition info
```

This line, built by several **echo -n** command lines and a final **echo** command line, indicates the progress of the remote operations. Once the collect script finishes, directory /admin/barb on cuthroat contains a copy of barb's system-administration files.

I could, of course, run collect for an arbitrary number of systems as follows:

```
cd /admin
for i in anthrax barb ducktape
do
collect $i
done
```

After collect executes in the example above, cuthroat's /admin directory is shown in Figure 1.



I can run collect on cuthroat to copy cuthroat's own files (rather than a remote system's files) as shown in the following example:

```
logon to cuthroat
cd /admin
collect cuthroat
```

If cuthroat's .rhost file names itself, the collect script will execute correctly and copy the collected files into cuthroat's /admin/cuthroat directory.

Example 1

If a disk failure were to obliterate one of my machines, the collected system-administration information would help me reload Linux with a minimum of confusion and difficulty on the replacement disk.

If loyd's disk failed, for example, I would replace the disk and restore Linux with these steps:

1. Reconstruct the partitions from information in /admin/loyd/fdisk.
2. Reload Linux.
3. Rebuild the kernel from the information in the file /admin/loyd/kernel.config.
4. File /admin/loyd/lilo.conf contains the information that the line **append="cdu31a=0x340,5"** is necessary for the proper operation of loyd's ancient CD-ROM drive.

There are, of course, as many deviations from these steps as there are users of Linux, but the point of showing the steps is to demonstrate how the collected information is useful in restoring a Linux system.

Although the ability to recover from catastrophic errors was the initial impetus for creating the collect script, the collected data has a number of other uses as well.

Example 2

Recently I needed to add an IBM Token Ring Network 16/4 Adapter to barb. This adapter only works with interrupt request (IRQ) 2, 3 or 7, so I examined the /admin/barb/interrupt file and determined that IRQ 3 was unused. Since I had collected this information remotely and stored it on cuthroat, I established that barb had an available IRQ without a trip to barb's location and without logging on to barb. In fact, since barb's information was stored on cuthroat, I could have located an unused interrupt for barb even if barb were off-line.

Example 3

Suppose I need to inventory some software or hardware component in each of the various Linux systems. Let's use networking cards as an example:

```
cd /admin
egrep -i "ne2000|3c|ibm tr" \
        `find . -name interrupts -print`
```

The egrep command will search the interrupts file in each Linux system's directory for ne2000 (the NE2000 clones), 3c (3Com cards), or ibm tr (IBM Token Ring cards) and print all matching lines in each file.

Example 4

Several months ago I configured the Enhanced Real Time Clock (RTC) support into loyd's kernel. Or was it speed's kernel? Could I have configured RTC support into both kernels? Here's how to tell which kernels have RTC support:

```
cd /admin
grep CONFIG_RTC=y \
    `find . -name kernel.config -print`
```

In a fraction of a second, grep confirms that only loyd has RTC support:

```
  ./loyd/kernel.config:CONFIG_RTC=y
```

Example 5

The cuthroat machine has a PC DOS partition. Recently I booted DOS on cuthroat to configure the autoexec.bat and config.sys files so that I could use cuthroat's CD-ROM under DOS. The instructions told me to take one action, if the CD-ROM were controlled by IRQ 14, and to take a completely different action, if the CD-ROM were controlled by IRQ 15. Being efficient (or lazy) I didn't want to turn off cuthroat, rip it open, determine where the CD-ROM cable plugged into the IDE controller, reassemble it and turn it on again.

After pondering a bit, the answer occurred to me: look at a copy of cuthroat's /proc/interrupt file which was stored on loyd. I didn't even have to boot Linux on cuthroat. I used a DOS FTP client to transfer loyd's /admin/cuthroat/interrupts file to the DOS system on cuthroat. Here are the two relevant lines from that file:

```
14:  9663 + ide0
15:    32 + ide1
```

IRQ 14 is the first IDE device; at the time the collect script obtained cuthroat's system-administration information, there had been 9,663 interrupts on this device. During the same interval, the second IDE device, attached to IRQ 15, had

generated only 32 interrupts. Since I knew cuthroat had only two IDE devices, it was obvious from the interrupt count that the hard drive was attached to IRQ 14 and the CD-ROM was attached to IRQ 15.

Example 6

As a final example, let's find all the Pentium processors with Intel's infamous floating-point-division bug:

```
cd /admin
grep fdiv_bug `find . -name cpuinfo -print`\
         | grep yes
```

If the Pentium chip in “solo” had the floating-point-division bug, then grep would produce the following output:

```
./solo/cpuinfo:fdiv_bug        : yes
```

Redundant Copies

Although cuthroat is my primary system-administration site, I keep the collected files on several systems for redundancy. After copying the system-administration information from all the Linux sites to cuthroat, I propagate the collected information from cuthroat to another system:

```
rsh loyd mkdir /admin
rcp -pr /admin/* loyd:/admin
```

I repeat the rcp for each machine on which I wish to have a copy of this information.

Requirements

Several simple requirements must be satisfied for the collect script to work:

- The first (and most obvious) requirement is that all systems must be interconnected.
- Depending on how name resolution is configured, all system names must be in a Domain Name Server or in each system's /etc/hosts.
- Each system needs a properly configured .rhost file to support remote shell and remote copy operations.
- And finally, you must configure the /proc file system in each system's kernel. Note that the kernel build procedure includes the /proc file system by default.

Extensions

The collect script can be easily extended if you find that /proc (or any other directory) contains system-administration information that is important to you. None of my systems use PPP; if yours does, modify the collect script to capture your PPP configuration information.

Most of my Linux systems run the Apache web server, but I don't bother to collect any Apache configuration information because only two lines distinguish one system's configuration from another. If you're running a web server and you've made a significant number of configuration changes, you may wish to collect your web server's configuration data.

If you are using Linux as a firewall, modify the collect script to save the firewall configuration. If Mr. Hughes had been using the collect script, the failure of his firewall's hard disk might not have cost him “hours of work time and probably a day of uptime”.

Running **find** on one Linux system located about a dozen files with names in the form *.conf. If you look at your systems closely, you may find additional configuration files to collect using the collect script.

Security Considerations

All of the Linux systems named in Figure 1 are protected from the Internet by an industrial-strength firewall. None of these systems are mission critical. My security considerations are probably quite different from yours, so you will have to evaluate whether any information you collect could compromise your systems and act accordingly.

Conclusions

The collect script simplifies remote system administration of disparate systems by centralizing configuration information. It is easy to use and easy to extend. Since the collected file sizes sum to less than 10KB per system, very little disk storage space is required. Although I created the collect script to ease recovery from potential catastrophes, the information obtained by using the collect script has a number of other uses as well.

Credits

Dan Lasley works on a large mass of Linux systems and a much larger mass of UNIX systems in his role as a Systems Engineer for Promus Hotel Corporation. When not fiddling with Linux, he is often observed hiking or taking photographs. In an example of multi-processing, he has even been observed doing both simultaneously. He can be reached via e-mail at dlasley@promus.co.

Archive Index Issue Table of Contents

Advanced search

Using Linux to Teach Unix System Administration

Joe Kaplenk

Issue #44, December 1997

A teacher at the College of DuPage in Illinois, has found that teaching system administration to beginners is both easy and cost-effective using Yggdrasil Linux.

College of DuPage

There are several handicaps inherent in teaching a class in Unix administration in an academic environment. In order to learn to be an administrator you need to actually administer the system—you must have complete control of the box. In most colleges and universities this is an impossibility, given security concerns, administrative overhead for such a setup and the need to equip individual workstations with the Unix operating system. Those colleges having such a setup in place have found it inefficient—either the hours must be limited or the machines go unused outside of class hours.

I have been able to alleviate the above concerns by utilizing Yggdrasil Linux, with the added advantage that the students can take the operating system home, work with it and bring it back to class for labs or assignments. Yggdrasil Linux can be run entirely off CD-ROM and RAM, and it uses a single diskette for booting and data storage. The overhead for such a setup is the same as the

normal maintenance costs for a PC; the only additional requirement being a CD-ROM drive, if one is not already installed on the system.

The College of DuPage in Glen Ellyn, Illinois (the second largest community college in the U.S.) has offered classes in Unix administration since 1985, due to the close proximity of Bell Labs and the former Western Electric facilities. Initially, the students ran Minux on an 8086 PC, and the lab was entirely devoted to Unix.

This lab was dropped when the College obtained an AIX system, and as a result, the students no longer had a place to practice administering a Unix system. Over time, more and more restrictions were placed on access to administrative files or doing administrative type functions not requiring root access. At first, access to the AIX system was over a network using a boot disk and, more recently, using a menu selection in DOS and MS-Windows. The overwhelming majority of students in the classes are computer professionals. The prerequisites for this class include “Introduction to Unix”, which in turn requires knowledge of a programming language.

When I started teaching the Unix administration class about six years ago, we were using a four-year-old book which did not cover many of the current features of Unix. In addition, there was no easy way to conduct meaningful labs. I went through three more textbooks over the next five years while looking for a book that was relevant to the students. Most textbooks covered too many versions of Unix, which was confusing for many students.

My teaching options were:

1. Continue using the system as it existed and resign myself to the students being unable to practice administration.
2. Request that individual workstations be set up to run Unix. As this had not been done before, this option would have been difficult, as well as costly, to implement.
3. Investigate running Unix on a PC without affecting the current academic networking environment.

I felt the third option was the only viable one. There are several manufacturers of Unix or Unix-like operating systems for the PC, including SCO Unix and Solaris x86. However, none of them have the wealth of free software along with available source code that Linux has. In addition, there are no licensing or distribution restrictions on Linux other than the GNU-type copyrights. It is supported by many USENET newsgroups and many helpful people on the Internet.

Last year, in addition to AIX, I decided to give my students the option of running Linux from a CD-ROM—in fact, I encouraged it. I was not willing to require it until I saw the results of my experiment. It was an overwhelming success and most students welcomed running Linux at home, since it allowed them to do assignments without having to come to the campus except for class. Only one student had a PC at home without a CD-ROM drive, and he was able to come to campus to use a PC. Therefore, this solution did not prevent any students from doing their assignments.

This year I made running Linux a requirement and chose as the textbook *Linux System Administrator's Survival Guide* by Timothy Parker, Ph.D., published by SAMS. I chose this text over a number of other books because it covered most of the class topics, was easy to read and was easily supplemented. I have written a *Lab and Projects* manual that includes assignments in Linux, AIX and generic Unix. I am planning to expand it to a full-fledged text as time permits, using recommendations of my students and my experiences.

Why Yggdrasil Linux?

My requirements for a Linux distribution for my class are that it be easy to run, require no configuration by the students and run right out of the box. I experimented with a number of versions of Linux. Except for Yggdrasil, all the versions either used the CD-ROM for installation only or had a live file system that required both a knowledge of Linux and the setup of a hard disk partition. We could fulfill neither of these requirements in our academic environment. Yggdrasil includes the boot diskette and does not require the student to configure either a boot or a root diskette.

The essential hardware ingredients are:

1. The kernel and other files are loaded into RAM from the boot diskette. This is mounted as the file system /ramdisk.
2. Boot Diskette contains the basic Linux kernel and basic Linux system files. Once the software is loaded into RAM, the boot diskette is no longer needed. This frees the diskette drive for loading customized files or backup.
3. CD-ROM loads more advanced files into /ramdisk and contains the remainder of the live file system, which is mounted as /.

You may think the missing ingredient is the hard disk, but it's not. I haven't listed the HD because you don't need it.

Linux is started by putting the boot diskette and Linux CD-ROM into the computer, then powering on or rebooting the PC. The PC loads the basic Linux

kernel and checks for various hardware devices by probing base addresses and evaluating the response. This probing process can cause the PC to hang, making it necessary to modify the boot parameters.

I have found that on many PCs it has been necessary or advisable to use additional boot parameters. The opportunity to modify boot parameters is given when the system displays the **boot:** prompt. If you wish to continue with no changes, you press enter.

You usually do not need additional boot parameters when running Linux on a SCSI-based system. However, I have taught in colleges which use IDE/ATAPI CD-ROMS on their PCS, not SCSI devices. Using an IDE or EIDE hard disk and CD-ROM presents some configuration challenges. Most current EIDE controller cards allow for primary and secondary IDE controllers. In addition, two devices, known as master and slave drives, can be attached to each channel. IDE CD-ROMs follow the same conventions as hard drives.

It is important that students understand this in order to set up their own PCs and use the college's equipment. Over half my students had problems that were eliminated once they understood the IDE configuration issues. My experience has been that it is better to have a motherboard with integrated, rather than separate, IDE controllers. Separate IDE controllers tend to have timing problems with some motherboards.

Linux Boot Parameters

I needed several common boot parameters in order to get Linux working on both the school's system and the students' home systems.

Figure 1. IDE Disk Layout in Linux

As shown in Figure 1, you can specify an IDE/ATAPI CD-ROM device by entering the “hd” device as follows:

```
linux hdc=cdrom
```

hdc refers to the /dev device and can actually be one of the following values:

- **hda**--the master drive on the primary IDE controller
- **hdb**--the slave drive on the primary IDE controller
- **hdc**--the master drive on the secondary IDE controller
- **hdd**--the slave drive on the secondary IDE controller

Some motherboards look at a single IDE controller as the primary controller, regardless of whether the board contains primary and secondary controllers.

Since most of the file system is stored in RAM, you need to increase the size of the RAM disk. On a system with 16MB of RAM, a RAM disk of 8MB is sufficient for classwork, leaving 8MB for the operating system. You can adjust the size upward as memory size increases. Even if you specify all the RAM as RAM disk, Linux is intelligent enough to reduce the RAM disk size to make space for the operating system. You set the RAM disc size with the following command:

```
linux ramdisk=8000
```

where 8000 refers to 8000 blocks of 1024 bytes each.

Another useful option is **hda=serialize**, a parameter that forces Linux to handle one controller at a time. A complete command line at the boot prompt could look like:

```
linux ramdisk=8000 hda=serialize hdc=cdrom
```

Other options can speed things up, such as disabling the check for a Sound Blaster Pro CD-ROM controller.

Getting to login

When you are done entering the **linux** command-line options, press enter. After all the messages detailing unsuccessful checks for various hardware devices, the system gives you a login prompt. The following four options are displayed on the screen:

1. **demo**--gives an X-windows demo program that runs very slowly from the CD-ROM on the first use, but runs much more quickly once it is in memory.
2. **guest**--sets up a guest login with normal user rights. This is useful for teaching labs in which students need to determine access rights for certain files and directories by accessing them as guest.
3. **install**--allows you to install Linux and gives several menus on the screen.
4. **root**--gives you permissions to all files and commands.

File System Structure

For the purposes of this discussion, a root login is assumed. When you type the **mount** command without arguments, the screen output will look similar to the following:

```
/dev/hdd on  / type iso9660
```

/dev/ramdisk on /ramdisk type ext2 (rw) /proc on on type /proc (type)

/dev/hdd refers to the CD-ROM mounted as root. **/dev/ramdisk** refers to the file system named /ramdisk, which is using internal RAM and can to be altered. **/proc** is the standard Unix pointer location for devices.

Figure 2. Initial Layout of Linux File System

I have my students enter the following command:

```
mkdir /ramdisk/mnt  /ramdisk/dos
```

This command creates two directories on /ramdisk to be used as follows:

1. /ramdisk/mnt is used to mount a Unix-formatted diskette.
2. /ramdisk/dos is used to mount a DOS-formatted diskette.

Creating a Linux File System on the Diskette

You format a 1.44MB diskette by typing the command:

```
fdformat /dev/fd0h1440
```

The following output will appear on the screen:

```
Double sided, 80 tracks, 19s sec/track. Total capacity is 1440
```

KB formatting...done verifying... done

Note that formatting will destroy any files already on the diskette you insert.

Now, to create a Unix file system on the diskette, type the following command:

```
mkfs /dev/fd0H1440
```

You can then mount the diskette by typing:

```
mount /dev/fd0H1440 /ramdisk/mnt
```

Now, when you type **mount** alone, the following output appears on the screen:

```
/dev/hdd on / type iso9660 (ro)
```

/dev/ramdisk on /ramdisk type ext2 (rw) /proc on on type /proc (type) /dev/fd0u1440 on /ramdisk/mnt type ext2 (rw)

Linux files can now be saved to /ramdisk/mnt. To unmount the diskette use the command:

```
umount /ramdisk/mnt
```

or:

```
umount /dev/fd0H1440
```

Figure 3. Layout of Linux File System with UNIX Diskette

Using an MSDOS-Based Diskette and Files

It is sometimes necessary to transfer student assignments from the Linux box to the AIX box. Because networking is not done under Linux, it is necessary to use the diskette as the transfer media. The transfer is done using FTP from a DOS/Windows PC to the AIX box.

You will need to format the diskette under DOS. (Be sure you unmount any diskettes from the above exercise.) DOS formatting is done in Linux with the command:

```
mformat a:
```

Mount the DOS diskette with the command:

```
mount -t msdos /dev/fd0H1440 /ramdisk/dos
```

Typing **mount** alone produces the following output:

```
/dev/hdd on / type iso9660 (ro)
```

/dev/ramdisk on /ramdisk type ext2 (rw) /proc on on type /proc (type) /dev/ fd0u1440 on /ramdisk/dos type msdos (rw)

You can now save files in DOS format to /ramdisk/dos. Note that you must use the standard DOS file naming convention of up to 8 characters for the prefix, then a period, then up to 3 characters for the extension. You can unmount the diskette with the command:

```
umount /ramdisk/dos
```

Figure 4. Layout of Linux File System with DOS Diskette

Printing from Linux

Printing is one of the biggest problems I have dealt with. Since setting up the print spooler requires modifying configuration files and running daemons, I opted for the “quick and dirty” until such time as the students were more experienced. Generally, Yggdrasil Linux sets up the PC printer port as /dev/lp1. For some reason unknown to me, it sometimes establishes /dev/lp0 as the printer port. Keep this in mind in the following discussion.

Files can be printed by using the following command:

```
cat filename > /dev/lp1
```

This sends the raw file to the device port.

```
echo ^v^l > /dev/lp1
```

Use the sequence **ctrl-vctrl-l**. This will send a form feed to the printer and eject the last page of your printout. The second step is necessary because without it the first step will not completely print the file. If you are logged in as guest, make sure you have read/write permissions by typing: **chmod 666 /dev/lp1** or **chmod 666 /dev/lp0**.

Some printers, such as the IBM Proprinter, have a problem printing Unix files using the above method, since they do not insert a carriage return upon receiving only the line feed at the end of a line. As a result, the output will step across the page. In this case you can use a Unix-to- DOS program such as **to-dos** to convert the file format before printing. Modify the printing command as follows:

```
cat filename | todos > /dev/lp1 echo ^v^l > \
```

/dev/lp1

The ultimate solution to the printing problem is, of course, to run the print spooler. However, if the configuration is not done just right, the spooler doesn't work; thus, using the “quick and dirty” way gets the students started. It also helps the students understand the printing process and learn how to debug printer problems.

Modifying System Files

The /etc directory is symbolically linked to the /ramdisk/var/etc directory; therefore, students have write access to most files needed to administer the system. The files are saved on the diskette in Linux format, and the next time the student reboots the system the changed files are loaded. It is even possible to modify the boot diskette; just be sure to back it up first.

Figure 5. Layout of Linux File System Showing /etc Linkage

After the above steps are completed and understood by the students, they can have full control of the Linux system. There is no need to network and no need to use local hard disks. I hope to include networking and related skills as my class progresses.

Yggdrasil Linux can be set up quite effectively to teach Unix administration. It is inexpensive, relatively secure and the students can use it at home without modifying their PCs. It helps solve the biggest problems I have experienced as an instructor of Unix. Once the hardware is in place and network security issues are addressed, I am looking forward to using Linux to teach networking as well.

Joe Kaplenk has been a part-time instructor at the College of DuPage in Glen Ellyn, Illinois for 13 years. He has taught classes in introduction to computers and programming languages as well as Unix. He works full-time for IBM CS Systems in Oakbrook Terrace, Illinois. This article is based on excerpts from a textbook he is writing for teaching Unix administration. Joe enjoys all forms of skating, writing, singing country music songs and country dancing. He also enjoys spending time with his wife Ramona and daughter Anisa. He can be reached at jkaplenk@aol.com.

Archive Index Issue Table of Contents

Advanced search

Linux Makes The Big Leagues

Sam Williams

Issue #44, December 1997

This report on the HP conference freatures the announcement by Hewlett Packard of a port of MkLinux to its HP PA-RISC machines.

It may seem odd to be reading about an HP conference in *Linux Journal*. The fact of the matter is that interest in Linux is growing world wide. Many hardware vendors are interested in Linux because, quite frankly, it scares them. Even more importantly there were a couple of events at this conference that underscore some very important trends in the computing industry.

Prior to the conference I was excited to attend a white paper presentation by Chip Richards (richards@eng.iac.honeywell.com) called “A Place for Linux”, which occurred on the fourth day. His discussion centered on the current uses of Linux at Honeywell. Currently, Honeywell is using Linux as a design platform for several applications that are then ported to HP-UX once the development is completed. This approach allows them to capitalize their investment in PC hardware while keeping development costs down. His presentation was well received by a group that I would estimate at approximately 100 people. The attendance was good considering the conference had approximately 1200 attendees.

There were three very interesting keynote speakers. The first was Stephen Wallach from Convex technologies. The second was Dr. Ira Goldstein of Hewlett Packard and the third was Robert Cringely, author.

The evening started off with Dr. Wallach's speech. Dr. Wallach is not only the head of Convex, but also its founder. He talked about the recent acquisition of Convex by HP and the advantages the merger would bring to both companies. Also discussed was the new HP Exemplar-class server based on Convex technology. Dr. Wallach is a very interesting presenter who clearly didn't have enough time for everything he wanted to say.

The second presenter was Dr. Ira Goldstein, Chief Technology Officer, CSO in HP. His presentation dealt primarily with the Internet and imagery. He discussed some new technologies for image production and transport that are being developed jointly by HP, Microsoft and Kodak. The presented ideas seemed sound and would minimize many of the performance problems that we have today when downloading images from the Internet. This new technology would not only improve the ability to download quickly, but also allow you to resize images dynamically with very little loss in quality.

The most astounding revelation came during Dr. Goldstein's presentation—he discussed a new port of the Linux operating system to the HP PA-RISC architecture. He and his development team had decided to port the Carnegie Mellon Mach kernel to HP PA-RISC in order to use it for their imagery work. Once the port was finished they began looking for an OS layer. Since HP-UX is bound tightly to its monolithic kernel, it was excluded from consideration. Knowing that last year Apple helped produce a Linux layer that would interoperate with the Mach kernel (MkLinux), they decided to go in that direction for their operating system. Goldstein's team went one porting step further and took the actual Apple Linux source code and ported it easily to the new HP environment. The results are a new port of Linux that will run on **any** PA-RISC based machine.

Goldstein went on to question the audience about how many people might be interested in this port. I would say that about 50 hands went up. The reality is that companies can't always replace their computing base after three years, even with hardware obsolescence. Those companies that cannot afford new hardware want to get the best usage possible from the machines they have. Sometimes this comes through the use of sources such as Linux. Additional information on this project and Hewlett Packard's sponsorship can be found at http://www.gr.osf.org/mklinux/.

The final presenter was Robert X. Cringely. Cringely is the author of the book “Triumph of the Nerds in Silicon Valley” that was recently produced as a PBS special for television. By and large, this was a very entertaining presentation. Cringely is a sleeper. He was introduced as a writer and TV personality. What we didn't find out until well within the presentation is that he is a computer consultant, that he was employee #12 at Apple, that he owns and directs a company building Internet-based software and that he is a very entertaining presenter. He discussed his beliefs for the near-term growth of the computer industry and answered many questions from the audience, sharing the stage with Steve Wallach for the Q&A session.

Afterwards, I had an opportunity to speak with both Dr. Goldstein and Mr. Cringely. Dr. Goldstein indicated that once his team had completed their work,

HP had been more than willing to freely give the port without formal support to the Internet community (see URL above). He also indicated that in the future if there is a good showing of interest in this port, someone from HP might upgrade the Mach kernel and Linux environment as changes arise. Dr. Goldstein also went on to suggest that many companies might be interested in this port to help them leverage their existing technology. While not necessarily interested in providing Linux, HP is still interested in keeping their workstations on the desktop as long as possible. The bottom line is that the workstation still says HP no matter what OS it is running. This concept provides a way for the company to keep the customer interested in the lean times when equipment budgets are low.

Sam has worked with Unix and its variants for over a decade. He has worked with Linux since the 0.95 days. Four years ago he designed one of the first mainframe to Unix conversion projects in the Midwest. When not trying to convert the masses to Linux, he spends as much time with his family (wife and 3 kids) as possible. Currently he is waiting acceptance into a Masters degree program for Computer Science. He can be reached via e-mail at samw@www.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The Quick Start Guide to the GIMP, Part 2

Michael J. Hammel

Issue #44, December 1997

This second article in the series introduces us to the basic features of the GIMP, a Linux power tool for the graphics artist.

Last month, in the first of this four-part series on the GIMP, we took a quick tour around the GIMP's installation and system requirements. This month we'll cover a few basics about how the GIMP works, which types are included and how they are used, and a bit about supported file formats. This information is somewhat condensed, and you may wish to read through it once, then read it again as you begin playing with the GIMP and its windows.

How Does the GIMP work?

The GIMP is a raster graphics tool, meaning that it operates on images as collections of individual points called *pixels*. Each pixel is made up of a number of *channels*. The number of channels depends on the image type being processed. For example, RGB (and RGBA) images consist of four *channels*, one each for levels of Red, Green and Blue and one called the Alpha channel which is used to determine the transparency for a pixel. Transparency for individual pixels becomes important when working with *layers*. Layers are a feature of the GIMP which permit an artist to create pieces of an image separately; the pieces can be manipulated on their own without affecting the rest of the image. This is useful, for example, when creating cover art for a magazine or CD. The image on the cover of the November issue of *Linux Journal* had many layers. The text for the word “Graphics” is made up of 4 layers, each combined with the layer below in a different way to create the final 3D effect. We'll talk about layers in more depth next month, when we cover the Image Window in Part 3 of this series.

The GIMP supports three types of image formats: RGBA, Grayscale and Indexed. RGBA was described in the previous paragraph. RGB images are just like RGBA except they do not contain an Alpha channel. Grayscale images are

similar to RGBA images except they only function with one channel—Gray—that allows varying levels of gray from black to white. Indexed images use a color palette to determine which colors are available, and each pixel's color is represented by an index value into that palette. For example, a value of 112 for a pixel means that the pixel will be the color specified by the 112th entry in the color palette. Each channel of an RGBA image uses 8 bits, allowing for 256 shades of each channel's color or transparency level. The grayscale channel uses 24 bits, supplying about 16 million shades of gray. Indexed images use 8 bits to define the color palette index, meaning a color palette consists of 256 unique colors.

Most processing work should be done in RGB (note that the GIMP usually refers to RGBA simply as RGB) or Grayscale modes to allow greater flexibility. Some image file formats, most notably the GIF format, save images as indexed images. If you open a GIF file for processing with the GIMP, you should consider first converting it to RGB. Afterwards, if a GIF file is required (say, for use on a web page), you can convert the image back to indexed. Conversion of image formats from within the GIMP is done using the Image Window menus, which we'll discuss at length next month.

GIMP Windows

When you start the GIMP, you'll find it provides a single, small window with a set of buttons, called the *Toolbox*. The Toolbox is the starting point for creating images and is made up of a menu bar, the tool buttons and the foreground/background color block. If you are familiar with Adobe Photoshop then the look and feel of the Toolbox should be quite familiar to you. Figure 1 shows the default Toolbox configuration.

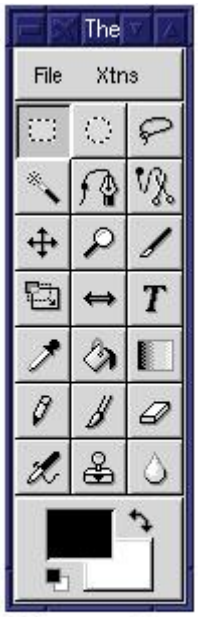


Figure 1 – Toolbox

The GIMP provides a number of other windows with which you should become familiar. The first is the *Image Window*, in which an image is displayed. There can be more than one of these windows open at a time. You can open the default Image Window by placing the cursor over the Toolbox and typing **ctrl-N** (N is for New Window). A small dialog box will open that allows you to specify some parameters for this new window. Accept the defaults for now so that you can see an example Image Window. These windows display an image as it will look in its final form based upon the layers that are currently marked as visible. Image Windows are resizeable and scrollable. It is possible, through the use of zooming features, to have an image that is larger than the display area of an Image Window as well as an image which is smaller than the display area.

Image Windows are made up of a number of smaller window features: rulers, menus and scrollbars. The rulers, along with their *guides*, provide a convenient method for determining location and size of areas of the image. All image windows have rulers on the left and top sides of the window, although these can be turned off using one of the pop-down menus.

The pop-down menus in the Image Window are not obvious—you have to hold down the right mouse button while the cursor is in the Image Window to open the menus (opening a menu is also known as “posting” the menu). These menus contain a wealth of options for viewing, selecting and manipulating the image. While many of the options are available directly from the Toolbox, many others, like the plug-in filters, are only directly accessible from pop-down menus. You'll probably want to familiarize yourself with the layout of these menus, since you'll be using them quite often.

The scrollbars appear on the right side and the bottom of the Image Window. They are used to move the image around the display area when the full image cannot be displayed within the current width and height of the window.

Dialog Windows

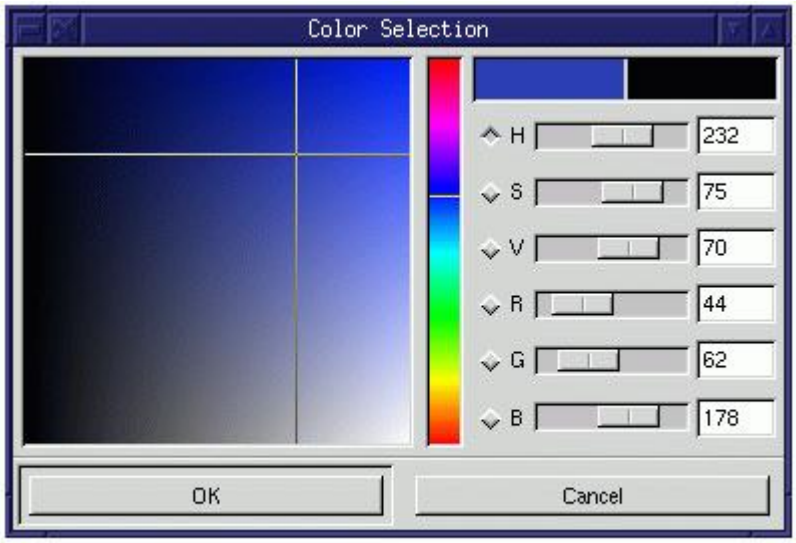


Figure 2 – Color Selection Dialog

Along with Image Windows, users will find that they interact repeatedly with various dialog windows. These windows open and close based on user input. For example, selecting New from the File menu in the Toolbox opens the dialog for creating a new Image Window. Most filters (features which manipulate all or part of an image) use dialogs to allow specifying parameters. Some dialogs require user action to close them; others are informational only and close automatically when they finish their work. An example of an informational dialog is the dialog box that shows the status of a filter operation. This dialog shows a scale that fills from left to right to represent the amount of the selected area that has been filtered.

There are a number of dialogs in the GIMP; a list of these dialogs is in the “Dialog Windows” box.

Dialog Windows

Cursors

The GIMP uses a number of different cursors to identify what can be done in different portions of the image when using different tools. A cross-hair cursor is used by the Selection tools. This cursor is also used by the Color Picker, Bucket and Blend tools. A double-crosshair is used by the Crop tool. A pair of curved, point-to-tail arrows is used for the Transform tool. A two-pointed arrow (arrowheads on both ends) is used for the Flip tool. The Text tool uses the

traditional I-beam cursor. The Move tool uses two two-pointed arrows, one pointing left/right and one pointing up/down. All other tools use a pencil cursor.

The cursor type also depends on whether a section of the image has been selected or not. For example, if a section of the image has been selected, and the cursor is placed over that selection then the cursor looks like the Move tool cursor. This is because the Selection tools permit a selected area to be moved without having to change tools to the Move tool. Note that only the selection moves—the non-selected region stays where it is. For many of the tools the cursor will be the traditional diagonal arrow over the image until the cursor is moved into a selected region. It takes a little work to become familiar with the currently active function based on the look of the cursor, but once you've worked with the GIMP for a short time they will become second nature to recognize.

Opening/Saving Files

Now that we've introduced the basic layout of the application we can start to look in depth. Beginning with file input and output, you have the option of starting with a blank new image or opening an existing image. In either case, you select the File pull-down menu from the Toolbox's menu bar. In this menu you'll find 5 options: New, Open, About..., Preferences and Quit. The About... option opens a little window that gives credit to the authors of various pieces of the GIMP. The Preferences option is used to set the visual cues used for transparent regions of images as well as the number of levels of undo. Keep in mind that setting higher levels of undo can use up significant amounts of memory. Changes to this setting are applicable to the current session only—they are not saved between GIMP sessions. There is an undo level option in the gimprc file, if you wish to make permanent changes. Finally, the Quit option does the obvious—it exits the program.

The New option opens a dialog that allows the user to select the dimensions of a new Image Window. The image type (RGB or grayscale) and the type of background to use are also selectable. Figure 3 shows the New Image dialog box. As with many options in the GIMP, you can also open a new window using keyboard accelerators. For a new Image Window, place the cursor over any GIMP window and type **ctrl-N**.

Figure 3 – New Image Dialog

Opening an existing image is similar to opening a new Image Window. The dialog box presented is the File Selection dialog, which enables you to change directories and select individual files for opening. If you've ever used a Motif or Windows-based application, you will be familiar with the way the File Selection window is used. Like the New option, the Open option can be accessed through the keyboard. Place the cursor over any GIMP window and type **ctrl-O** to open an existing image.

Supported File Formats

In order to open an existing image, you must be familiar with the file formats supported by the GIMP. Raster images can be saved in a large variety of formats, each suitable for various functions. The GIMP supports all of the more popular formats such as GIF, JPEG and TIFF, and a few lesser-known formats. The GIMP Plug-In API, a programming interface allowing users to add extensions to the GIMP, enables the list of supported formats to be extended. All that is necessary is for a user to write a plug-in to handle the reading and writing of the new format.

The list of raster formats supported in the default distribution for reading and writing includes those shown in the “Raster Formats” box.

Raster Formats

PostScript output is used in conjunction with a Print Plug-In. This plug-in is not in my distribution, but it will most likely be part of the basic package by the time this article reaches the newsstand.

Four GIMP-specific formats are also supported—GBR, HEADER, PAT and XCF. GBR is the format used for brushes, so that you can create a simple image and save it as a new brush quite easily. The HEADER format is used internally for

the buttons in the Toolbox. PAT files hold the patterns used for Bucket Fills and other fill operations. XCF is the format used to save layer information. While you are working on an image, you should periodically save it as an XCF image so that, if necessary, you can load it again in the future with all the layer information intact.

Saving images can be a little complicated. For example, if you try to save an image as a TIFF file you might find that only part of the image gets saved. When specifying any format other than XCF, the GIMP will save only the currently active layer unless you *flatten* or *merge* the visible layers of the image. If you save the image using the XCF format (even without first flattening or merging the layers), all of the layers will be saved. The moral here is simple—save your images frequently as XCF files, and you won't lose any data.

Some file formats are meaningful only with certain image formats. An RGB image contains more information about an image than the GIF format holds, so RGB images cannot be saved in the GIF format. If you wish to save the image as a GIF, the image must first be converted to an *indexed* format. To accomplish this, flatten the layers (done via the Image menu's layers**->**flatten option), click the right mouse button over the image to drop down the image menus and select image**->**indexed. The image is converted and is now ready to save as a GIF file. Note that converting from RGB to GIF means that some of the information in the original image may be lost, although the loss may not be visible. If you wish to enlarge or resize the image later, you should first save the flattened image as a TIFF file. Later you can convert it to an indexed image and re-save it as a GIF file. You might want to do this when working with web page graphics, for example.

Don't let all this confuse you. In short, start with these simple steps:

1. Create your work of art in the GIMP.
2. Save it as an XCF formatted file.
3. Flatten the image.
4. Save it as a TIFF formatted file.
5. Convert the image to indexed format.
6. Save it as a GIF formatted file.

At this point you have the original layers information in the XCF file, a full color, high quality image in the TIFF file and an image suitable for use on your web pages in the GIF file. Each of these file formats is a different size. The XCF is the largest; the GIF is the smallest. As you can see, working with images in this way tends to be very disk space intensive.

PhotoCD and Digital Cameras

Support for the PhotoCD format, available from Kodak Digital Cameras, was available in the 0.54 release of the GIMP last year. A quick check of the Plug-In Registry shows that the plug-in has not yet been ported (or at least not registered) for the 1.0 release. If you need this format, you should check with the plug-in author or check the Registry periodically to see if the plug-in has been registered. Note that plug-ins written for the 0.54 or 0.60 releases of the GIMP are *not* compatible with the 1.0 release. They must be rewritten using the GIMP 1.0 Plug-In API.

Vector Formats

The alternative to raster graphics is vector graphics. There are a wide variety of image file formats that the GIMP supports for raster images, but it does not support any image files that use vector formats. This is only important if you wish to use one of the many ClipArt collections available for Windows systems; these files are often suffixed with .wmf. There are still many CD's available with image collections in TIFF or JPEG format that can readily be used by the GIMP. If you must have access to the vector ClipArt collections, use Applixware's Applix Graphics package or perhaps a Windows program to read them in and save them in a raster format such as TIFF, JPEG or GIF.

PNG

An emerging format that has recently gained wide acceptance is called PNG, the Portable Network Graphics format. Earlier releases of the GIMP support this format but, at the time of this writing, the plug-in providing PNG support has not yet been ported to the new 1.0 Plug-In API. I expect this to happen by the time this article is published. Again, check the Plug-In Registry to be certain.

Moving On

We've covered a lot of ground but have only managed to open a file—we haven't done anything useful with it. Don't despair—we're getting there. Next month we'll cover the Image Window in more detail. A detailed discussion on layers will also be presented, which is very important to understanding how to get the most out of the GIMP. We'll briefly cover how filters work and discuss a few of the more interesting image-processing options available with filters. After that, in the final installment of this series, we'll cover the Toolbox in complete detail. At that point you'll have enough background to begin to do something really useful with the tools in the Toolbox.

I realize some readers will wonder why I don't cover the Toolbox before I cover filters. Well, it has to do with the length of each of these installments. The best

way to fit everything into reasonably sized articles is to do the layers and filters section together. Since the Toolbox has so many tools, it will take a very long installment just to give a short introduction to each of them. Stay tuned: in the end, having the information from all four parts will be of benefit as you make the most of what the GIMP has to offer.

Resources

A Computer Science graduate of Texas Tech University, ***Michael J. Hammel*** *mjhammel@csn.net, is a software developer specializing in X/Motif living in Denver, Colorado. He writes the monthly Graphics Muse column in* Linux Gazette, maintains the Linux Graphics mini-HOWTO, helps administer the Internet Ray Tracing Competition (http://irtc.org/) and coauthored *The Unix Web Server Book*, published by Ventana Press. His outside interests include running, basketball, Thai food, gardening and dogs.

Archive Index Issue Table of Contents

Advanced search

LJ Interviews Larry Augustin

Marjorie Richardson

Issue #44, December 1997

I did an interview with Larry Augustin, founder and President of VA Research, by e-mail on August 19, 1997.

VA Research was founded in 1993 to develop affordable workstation, server and Internet products using the Linux operating system. I did an interview with Larry Augustin, their founder and President, by e-mail on August 19, 1997.

Marjorie: Tell us how you first became interested in Linux.

Larry: It was during the time that I was a Ph.D. student at Stanford University in the summer of 1993. I did a fair amount of system administration work in our lab. I was also consulting and doing system administration at Fintronic USA, an ECAD software company. At the same time Linux was beginning to come into its own. By that I mean, the essentials such as X and NFS were supported. The Linux environment was pretty close to our primary development platform, Sun OS 4.1.3; so, we decided it was time to replace Sun with Linux. I set up a 66MHz 486 machine running Linux for around $2000 and ported my thesis (15,000 lines of C/C++ code and 300 pages of LaTeX) as the first test case. Even though my code was very Sun dependent, it took me less than a day to port it to Linux. Compared to the SPARCstation 2 machines we had ($7000 base price), the 486 running Linux ran my code 1.5 to 2 times faster. I was sold. By the way, we still use that first 486 at VA Research in Receiving.

Marjorie: Since you come from the West Coast which is traditional BSD and Sun OS territory, what factors made you decide to choose Linux over BSD?

Larry: Actually, my first exposure to Unix was System V Release 3 at AT&T Bell Labs. I spent two years there, beginning in 1984. So some of the System V

leanings of Linux appealed to me. One thing I liked about Linux was the way it combined the best of System V and BSD. For example, I liked the combination of System V run levels with BSD-style **rc** scripts used in SLS and later Slackware. I also tried FreeBSD on that same machine that first ran Linux. FreeBSD was extremely appealing because I was so familiar with Sun OS. At the same time, FreeBSD was missing important features like shared libraries, and it didn't have the user base or the developer base of Linux. It was also slower than Linux. The trend was all Linux, and I didn't see BSD catching up.

Marjorie: What made you decide to go into the Linux business?

Larry: James Vera, another Ph.D. student at Stanford, looked at the Linux machines we had put together and compared them to the SPARCstation we were using. They were 1.5 to 2 times faster, cost one-third as much, had more software and also ran DOS/Windows. We showed them off and people said, “Can you put one together for me?” Graduate students can always use more money and another excuse not to work on their thesis, so we decided to put together systems for people on weekends. The original idea was to gather all the components in my apartment and have an assembly party. The concept was simple, but it turned out to be a lot more complicated. As one of the venture capital people we talked to later put it, “the proverbial lemonade stand”.

The 1993 equivalent of the lemonade stand was the web page, which we ran out of Fintronic, where I did consulting work. Those first web pages were really ugly, but they contained a lot of information that appealed to people. We'd done our homework: benchmarked different configurations, tested different video cards, etc. We thought we had a pretty nice machine for the money.

Marjorie: Sounds as if you almost went into the web business rather than the Linux business.

Larry: We didn't necessarily plan to build a business around Linux workstations when we started our “lemonade stand”. It was originally a part-time business. We'd been doing it about six months and business was getting to be more than we could handle. During that time, two other friends of mine from Stanford, David Filo and Jerry Yang, were having the same problem with a little web site they had developed called Yahoo. James and I were working full time on Linux, and Dave and Jerry were working full time putting URLs into Yahoo. Not a lot of work was getting done at Stanford.

Dave, Jerry, James, myself and another friend, David Ku, began writing business plans for Internet-based businesses. We wrote a couple of business plans together, but none of them really clicked. We were each being pulled into the

separate directions we had already taken. Dave and Jerry ended up with Yahoo going public and making $150 million each. Now I tell people that my claim to fame was writing a business plan with Jerry Yang and David Filo and turning down $150 million to work on Linux.

Marjorie: When did you make your first sale?

Larry: It was in November 1993. It wasn't a complete disaster, but we did learn that being in the hardware business is not easy. Lots of things went wrong. First, we were using the Orchid Fahrenheit video cards based on the S3-028 chip set. The cards we received for that first order came with a different BIOS revision than the cards we had tested and didn't work with X, and the multi-I/O cards we had tested were out of stock. We eventually got things worked out, but it was not as easy as we had thought it would be.

The process we followed with all our components was fairly straightforward. We'd get samples of 3 or 4 of the best-looking products in a category based on our experience and comments on the Internet. Then, we would test and benchmark them and use the best ones. But it's not always that easy. First, manufacturers are always doing product revisions like the BIOS change. When you're doing only a few parts a month, it's really hard to get the same revision of anything. Manufacturers also change revisions and don't tell you. We had that problem again recently with Matrox. The Matrox Millennium used to work in 24-bit color mode with XFree86, then Matrox did a BIOS upgrade. So, we had to get the information on the new BIOS back to the XFree86 core team.

We also discovered there is a lot of bad cache and RAM out there. We now qualify each brand of SIMM or DIMM down to the specific chips used with each motherboard. We used to think you could buy any quality brand memory and expect it to work with any quality brand motherboard—that's not true. We now use only memory qualified with each board.

In general, for a given manufacturer, quality differs by manufacturing lot. For example, we'd have a disc drive that looked okay, and then receive a batch with 50% failure rates.

We ended up developing a whole set of checklists and test procedures for every system. Benchmarking each system is also important since some errors only show up as performance problems. Things have also gotten easier, as we have grown. We can now qualify batches of 50 and 100 components all from the same manufacturing lot. That helps somewhat, but one thing we have learned is that there's a lot more junk out there than we thought.

Marjorie: What do you think are the key barriers to acceptance of Linux in the corporate world?

Larry: First, I think that by “acceptance” we mean acceptance by corporate MIS departments. Most of the Fortune 500 already use Linux but corporate MIS doesn't know it. The feedback we get from the traditional magazines like *UNIX REVIEW*, which are primarily Fortune 500, is that Linux is well established among their subscribers. Usually, it's because a developer or system administrator snuck a machine past the corporate people. Eventually, the MIS people discover the machine because some application is costing less and running better. This strategy of sneaking it in the back door seems pretty common.

For example, a good story I heard recently was from Cisco. Cisco has 10,000 people and 1600 printers worldwide. Two years ago they had a horrible management problem with printers. They could never tell what was happening with a networked printer, since hundreds of people on dozens of systems could all spool jobs to that printer. The system administrator responsible for printers began designing a scheme where all print jobs for a given printer went through one spooling machine. The spooling machines needed to be able to talk to all kinds of clients (Windows, NT, Unix, Novell, etc.); they needed to be inexpensive enough that they could deploy hundreds of them; they needed to be reliable; and they needed to be remotely administered. Now all of those 10,000 people around the world can print to any one of those 1600 printers anywhere else in the world, and all those print jobs are spooled through Linux machines.

Marjorie: Continuing that same subject, what are the barriers to corporate MIS?

Larry: Support is a big one. Corporate MIS wants someone to take responsibility. We all know Linux has incredible support through the Internet, but that's not the same as having someone's name on a legal document guaranteeing support. That's where the commercial Linux vendors like Caldera, Red Hat and VA Research come in—the MIS people need to see a large stable company that will be there to back them up.

Applications is another big one. Ninety-five percent of the corporate world doesn't care what OS the desktop runs, but whatever OS it is, they want it to run the latest version of Microsoft Office. You can see the power of Microsoft Office in the recent deal Apple struck with Microsoft.

Perception is one of the barriers I don't think people give enough weight to. The corporate world has to perceive Linux as friendly to corporate MIS people. The perception that Linux is for hackers only has to change. People in the Linux world need to project a courteous and professional image.

Marjorie: How does VA Research support the Linux community?

Larry: First, we host the Silicon Valley Linux Users Group (SVLUG) and the Debian project on machines at VA. That means we provide them with machines and network access. We also sponsor SVLUG in other ways, such as providing space for install fests and workshops. Usually, you'll find a couple of VA people at the install fests helping out.

Also, we maintain a fairly complete FAQ and related support information on our tech support web pages. These are available to anyone, not just our customers.

Archive Index Issue Table of Contents

Advanced search

OmniBasic

Eric Harlow

Issue #44, December 1997

The package includes some small sample programs and a small, yet fairly complete, manual that references the OmniBasic language.

- Manufacturer: Computer Design Lab
- Phone: 573-236-4644
- URL: http://www.bmtmicro.com/catalog/omnibasic.html
- Price: $89 US (per copy)
- Reviewer: Eric Harlow

After having installed several compilers that consume 50 megabytes of disk space, it was a relief to install a compiler that was this small—less than one megabyte. The package includes some small sample programs and a small, yet fairly complete, manual that references the OmniBasic language.

The OmniBasic compiler runs across several platforms, and the programs written for one should compile on the other platforms assuming that you do not have any platform dependencies.

Basic has evolved since the days of line numbers on every line, and this product has evolved too. The OmniBasic language is a “structured” basic language that is mostly backwards compatible with the old, line-numbered programs (just in case you need to run one of those stored on your cassette tape). The language features subroutines and functions with parameters, structured loops, file I/O, built-in string-manipulation routines (RIGHT$, MID$, LEFT$) and math functions. It also has the ability to manipulate pointers and access system functions. For backwards compatibility and for people with poor coding techniques, the language also contains the GOSUB and GOTO statements and supports line numbers.

The language takes the approach of the gnu FORTRAN compiler by converting the BASIC code to C and letting the gnu C compiler finish up the work. As a result, the programs are fast and compact, although not as small as straight C code. The OmniBasic compiler will show the output as C or assembly language, and C code can be mixed with BASIC.

OmniBasic has recently added GUI support using XForms. The beta version I tested worked well, and the release version should be out by the time you read this review. The GUI support is also expected to be cross-platform.

Summary

OmniBasic is a small, well-documented package for Linux. The language is small and easy to pick up. The addition of GUI support makes it an easy language in which to write those quick graphical interfaces.

Eric Harlow has been running NetBrain on Linux since February 1996. He's currently a consultant at RDA Consultants Ltd. His e-mail address is brain@netbrain.com.

Archive Index Issue Table of Contents

Advanced search

BRU 2000 for X11

Garrett Smith

Issue #44, December 1997

BRU 2000 for X11 is targeted for users who want to easily back up their computer without having to learn the command-line interface for their backup software.

- Manufacturer: Enhanced Software Technologies, Inc.
- E-mail: lj-info@estinc.com
- URL: http://www.estinc.com/
- Price: $89 US (personal edition) Free for xbru only
- Reviewer: Garrett Smith

While not the most flashy area of computing, proper backups are essential. BRU 2000 for X11 (**xbru**) is a user-friendly graphical interface for EST's BRU backup software. It is targeted for users who want to easily back up their computer without having to learn the command-line interface for their backup software. To use the more complex options of BRU 2000, you still need to learn BRU's command-line interface.

xbru came with BRU 2000; however, I downloaded the newest version (as of July 1997) of xbru from EST's web site. **xbru** is freely available, but it does require BRU 15.0 which is sold by EST Inc. **xbru** also requires Tcl 7.6 and tk 4.2, which are freely available (see Resources).

I had one problem during the installation procedure for xbru—it wanted to install its files into the /usr/local/lib/bru directory, but, unfortunately, on my computer, /usr/local is mounted read-only from the NFS server. I sent e-mail to support@estinc.com asking how to install into a directory other than /usr/local/lib/bru. I later got a reply to my question, but before the reply came, I had obtained permission to install xbru on the NFS server.

Once I had root access on the server, the rest of the installation was straightforward. I ran **bru**, which was already installed, on the X11 package from the command line using the following command:

```
bru -xvf bru4X11.bru
```

Executing this command produced an install script to do the installation, which I then ran.

Figure 1. Main Page of BRU

When you start up xbru, it asks you which device you wish to use for backup. You can chose from the devices in your /etc/brutab file, or you can select a file. I used a file for testing at first, and later I tried it on a 4GB SCSI 4mm DAT tape drive. The main screen gives a few default options that make standard backups very easy: Full, Level 1 and Level 2. If you are root, Full does a full backup of the entire system; otherwise, it does a full backup of your home directory. The Level 1 and Level 2 options are more complicated. Level 1 backs up files modified since the last Full backup. Level 2 backs up files modified since the last Level-1 backup.

Help

BRU comes with a hefty manual for its command-line software which I didn't have time to look through thoroughly. There is a 25-page booklet for the X11 interface that is very basic and generally easy to understand—a little more detail might have been helpful. There is context help for the backup and restore operations which was helpful, but would have been even more helpful if it had included other operations and dialog boxes.

The main on-line help was more troublesome—it launched a web browser (Netscape, by default) to look at EST's web page. Starting the browser suspended xbru until you exited the browser. As a result, you can't easily refer to the help page while using xbru. You can start a web browser independent of xbru and go to EST's page, but, of course, this solution is less convenient. Also, xbru points the browser at EST's main web page, not a page dedicated to help for BRU and not to a local help file. This last can be particularly bothersome, if you are not always connected to the Internet.

Problems

After resolving my problems with the /usr/local directory, I had no other major problems. Mainly, xbru just showed a general lack of polish—as if it had been

rushed out the door in order to be able to advertise the inclusion of a user-friendly GUI. Some of the little problems I had were:

1. I received several Tcl error messages when xbru got confused.
2. **xbru** locked up every time I did a search through the backup listings.
3. Whenever I created a new backup, the buttons to restore, verify or list the backup were disabled until I restarted the program.

User Interface

The whole point of xbru is to supply an easy-to-use front end for BRU, so the interface is a big part of the package. Overall I liked the interface. The front screen was simple, with only a few options. Unfortunately, the Full, Level 1 and Level 2 buttons are slightly confusing to a new user who hasn't looked at the documentation closely. The little message bar at the bottom of the screen does explain the buttons more thoroughly; however, since there is a lot of unused space on the front screen, labeling the buttons “Full Backup”, “Level 1 Backup” and “Level 2 Backup” would be less confusing. Also, substituting an explanatory paragraph next to the buttons for the large EST logo would certainly be more helpful to the new user.

Figure 2. Backup Screenshot

The backup page was similar to other backup programs I have used, and I found it generally intuitive.

Figure 3. Restore Screenshot

The restore page was not quite so easy to understand. The confusing thing about the restore page was that on the left it gave a listing of files labeled as the current directory, but it listed only those files I had backed up, not all of the files on my hard drive. I could move these files to a listing labeled as the backup device to restore the files. While it worked as I expected from using other backup products, the labels just seemed confusing.

The verify and list operations were relatively simple and understandable.

Extra Features

Along with the ability to do backups on the spot, you can also save backup definitions and schedule them to be run at any time. This ability is essential for any long-term backup strategy. The xbru interface for this option was very easy to understand. Another nice option that came with xbru was a utility to configure the /etc/brutab file. This is beta software so it isn't mentioned in the

documentation. Hopefully, by the time this review is published, a final version of this software will be available.

I would recommend xbru for someone who wants to do backups of a home computer but nothing much more complicated. (Don't get xbru confused with BRU 2000's command-line interface which is quite powerful and robust—see below.) It is easy to use and doesn't require you to go digging through lots of documentation before you use it. I managed to use nearly all of its features without having to look in the manual.

About Command-Line BRU

xbru is an interface to the original, command-line BRU backup software. BRU is a very flexible and powerful backup program. It is much more reliable than tar and friends, supporting CRC error detection and automatic verification after backup to ensure that you don't have a bad tape. You can schedule backups to run automatically and even change the tapes in an auto-loading tape drive for multi-tape backups. BRU backs up all types of files: device files, sparse files, pipes, soft and hard links, etc. The list of features is too long for me to cover here. For a full review of BRU, see “BRU—Backup & Restore Utility”, *Linux Journal*, March 1995, Issue 11.

Resources

Garrett Smith (gsmith@univprep.pvt.k12.wa.us) is a summer intern at SSC and a student at University Prep. He enjoys telemark skiing, running and road biking. Computer related interests include working on **mnemonic** and trying to get Linux installed on as many computers at his school as possible.

Archive Index Issue Table of Contents

Advanced search

Raima Database Manager++, Velocis Database Server

Nick Xidis

Issue #44, December 1997

The most unique feature of the Raima line-up is the combination of pointer and relational data navigation.

- Manufacturer: Raima Corporation
- E-Mail: sales@raima.com
- URL: http://www.raima.com/
- RDM++ Price: $995 US (royalty free)
- Velocis Price: $495 US (2 users)
- Reviewer: Nick Xidis

Raima Corp. offers three products: the Raima Database Manager++ (RDM++, formerly db_Vista), a database management library for C, Raima Object Manager(ROM), a persistent class library for C++, and the Velocis database server, which supports both the RDM++ API and ANSI SQL. All three support transaction logging and recovery, compound keys and time stamping. The most unique feature of the Raima line-up is the combination of pointer and relational

data navigation. I tested all three using Caldera's OpenLinux Standard 1.1 on a DEC P-90 with 40MB RAM. I'll cover the RDM++ and Velocis server in this review.

RDM++ Installation

The installation is a snap. Just unpack the two floppies using **tar** and run the **makeall** shell script. RDM++ compiles with no significant errors. I had one minor gripe, the makeall shell script should ask for extra compile flags; I would have liked to add a **-O3** flag.

The RDM++ API

To create databases, begin with a .ddl file that defines the schema for the database. Tables are defined in a format that is very similar to STRUCT in C. Also, defined in the .dll file are sets. Sets define pointers to other records in the database. So, one record in a table can point directly to another without joining. This combination of tables (relational data model) and sets (network-pointer data model) is where RDM++ really shines. Using sets to point to other records is like having a permanent predefined **join** between tables and can speed up your application compared to the process of iterating over tables to create joins at runtime. In practice, I found that having joins built into the database produced shorter and cleaner code than similar applications using ISAM libraries. Once the .ddl files are built, the **ddlp** utility compiles a database dictionary and header files that supply constants needed by your application from the database.

The C API for RDM++ is a snap to learn, as all the function calls start with d_ and are very intuitive. To open a database, just call **d_open**; to read a record, **d_recread**. Don't think that because the API is intuitive that it's not rich—there are over 200 functions. The learning curve is very short; I was able to produce simple C applications within a couple of hours. One of the biggest aids in learning the API was the **dal** utility that allows d_ calls to be entered at the command line, just as they are written in sources. Almost any of the d_ function calls can be run interactively, and the results can be seen right away.

Another great time saver is the db_QUERY utility. This allows reports to be produced from RDM++ databases using SQL. Both tables and sets can be accessed from db_QUERY—sets appear as SQL views. You can use it either interactively or with canned query files. The neatest capability is that you can embed db_QUERY inside C applications. I used it at the command line to work out my queries. Then with a few function calls the same report is generated inside of my C application. For simplicity, the functions all start with q_ and there are only twelve of them.

I found this whole system of creating a database easy to understand and a joy to use. The only piece missing is a GUI report writer. Raima makes a fairly good one for MS Windows, but not for X.

RDM++ Runtime

Single-user applications can just be executed; however, to use the RDM++ multi-user mode, a lock manager is required. The lock manager runs in a separate process and manages the file locks for all the users of your application. The multi-user mode worked well. Record-level locking is not enforced by the lock manager, so you'll have to limit the number of users to 20 or less.

This is the one area in which the RDM++ falls short. If serious multi-user applications are going to be built with RDM++, you need record locks. I realize that there is some overhead with this option, but it seems well worth the cost.

Velocis Server

Velocis is a back office power house. It supports all of the standard stuff that you'd expect in the best commercial database servers: transaction processing, ACID transactions, record and table locks, stored procedures, triggers and hot backups. It also supports a wide set of APIs: RDM++ C API, ROM C++ API (for storing C++ objects), ANSI 89 and most of the 92 SQL API and SQL Access Group's Call Level Interface (SAG-CLI, which is the same as Microsoft's ODBC level 1 API). In addition, the server is easily extended with C or C++.

Installation

Installing Velocis was fairly easy. Once the disks are unpacked using tar, run the **install** script which creates two files: rdshome.sh and rdshome.csh. These shell scripts export all the environment variables that Velocis needs to run. I copied the contents of rdshome.sh to /etc/profile so that it's available system wide. Each Velocis server (you can run multiple servers) must have a host alias and a port of the same name defined in the /etc/services file. Just fire up the server and you are ready to get started.

Administration

Velocis is administered from the **rdsadm** command-line utility. Velocis is fairly easy to use and understand, but the rdsadm utility is cumbersome and doesn't do justice to the quality of the server. Raima should supply an X windows GUI administration tool with Velocis.

Unique Features

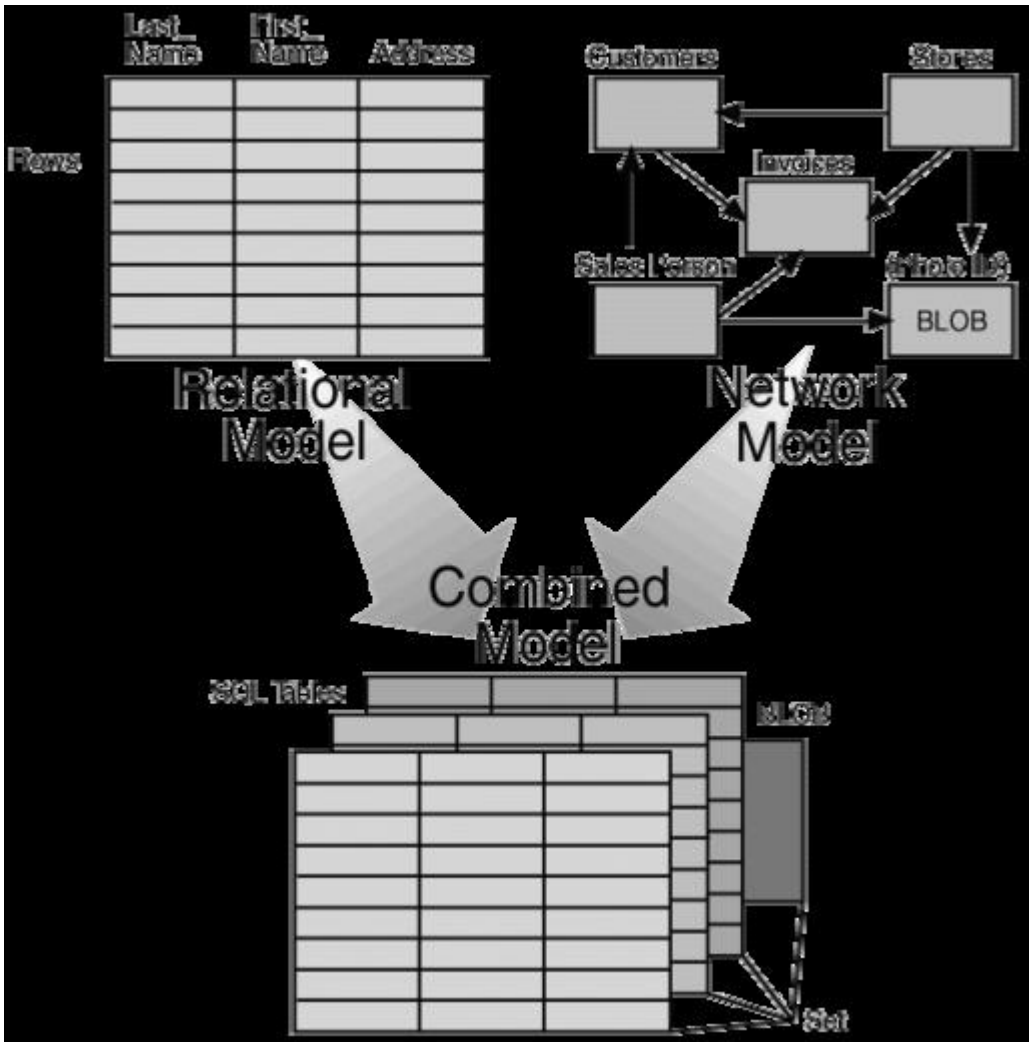
Velocis is a dream for C programmers because the server extension modules (EM) and user-defined functions (UDF) are all implemented in C.

UDF are C shared libraries that contain scalar or aggregate functions that can be called from SQL. UDF use the standard SAG-CLI and are registered by using the SQL **create function** call. Once registered, UDF can be used like any other SQL function.

UDF can be used in conjunction with the SQL **check** clause to create triggers. Triggers run on the server whenever a table is changed. Thus, by using the **check** clause when tables are created, UDF and built-in functions can be executed automatically on the server. This is a great way to ensure that custom business logic is enforced on your database.

The Velocis server can also be customized outside the SQL system with extension modules. EM are essentially the same as user-defined functions but are invoked by clients directly using RPC calls. They are independent of the SQL system. As a matter of fact the whole SQL interface is an extension module. This modular design makes Velocis an excellent choice for vertical market applications where heavy customization is required.

Velocis also supports stored SQL procedures with the **create procedure** call, allowing a group of SQL statements to be executed with a single function call.



Conclusion

Raima's products are all winners. I especially like the RDM++ API. The .dll set up and the d_ functions are very easy to learn and are rich enough in features to be taken seriously. Even with Velocis I'd be tempted to use the low level d_ functions in place of SQL. SQL is still nice to have for reporting and system administration. It is, after all, almost the universal database language. Raima only misses on a couple of counts. First, the lack of GUI tools for Velocis, an administration tool and report writer for X Windows as a minimum are required for serious back office use. Second, Raima needs to beef up its Internet tools. Currently, there is only a Perl interface for Velocis. Raima should consider doing a JDBC type 4 driver and an HTML scripting tool like PHP/FI or Scriptease for ADABAS D. All in all Raima is a winner and worth a look for your next project.

Nick Xidis is a Telecommunications Specialist with the Federal Aviation Administration in Auburn, WA. When he's not at work with the FAA, Nick does private consulting for Linux/Unix systems. He also enjoys playing with his wife and five (soon to be six) children. He can be reach via e-mail at nickx@xsc.wa.com.

Perforce Software Configuration Management System

Tom Björkholm

Issue #44, December 1997

The Perforce package goes a long way toward meeting those goals. In the sales material, Perforce is said to be, “Easier to use than Kleenex, runs faster than a red Porsche ...” Let's check if this is a valid claim.

- Manufacturer: Perforce Software, Inc.
- E-mail: info@perforce.com
- URL: http://www.perforce.com/
- Price: $500 US (free for non-commercial use)
- Reviewer: Tom Björkholm

Many of us usually talk about “revision control systems” when we actually mean software-configuration management systems. Most of us do have mixed feelings for these systems. We want the benefits they give us: tracing, branches, stepping back to an old version and other such options. However, we do not want to pay the usual price of overhead, complexity and extra work to set up and use the system. What we really want is a fast, easy-to-understand system that gives us all the benefits without any of the grief.

The Perforce package goes a long way toward meeting those goals. In the sales material, Perforce is said to be, “Easier to use than Kleenex, runs faster than a red Porsche ...” Let's check if this is a valid claim.

Installation

The Perforce system is a client/server system with just two binaries. The **p4** binary is the client program. On Linux/Intel the p4 binary is just 55KB. The server program, **p4d**, is just 388KB. These files are the only ones you need. However, it is also a good idea to get the man pages p4.1 and p4d.1, as well as the manual (a PostScript file). You can find all of these files at http://www.perforce.com/.

The server program, p4d, is started like a daemon and does not need to be run as root. Personally, I have found it to be a good idea to create a special user, called **Perforce**, that runs the server program in its home directory. This user is not necessary, but it certainly helps to keep the system organized.

Installing Perforce is easy. First, the client program needs to be copied to a location that is in the default path (for example, /usr/local/bin). Then, the server program **p4d** needs to be started by the rc files at system startup and stopped at system shutdown. The programs do not require anything special in their environment.

Basic Ideas

Perforce is based on a true networked client/server approach. To do their work, the client programs connect to a TCP port on the server. Thus, you only need a TCP route from the client to the server. There is no need for an NFS-mounted file system.

The user does all the work (such as editing and compiling) on local files. These files are kept in syncronization with the server by a few simple p4 commands.

Each user of Perforce is a client and has a client view that is stored in the server. The client view is used by the server to determine which files in the server database the client wants and what local names the client wants for those files. The client view can be changed using the **p4 client** command. The server's database of files is called the “depot” in Perforce.

An interesting idea in Perforce is the notation of a change. A new numbered change is created each time you submit one or several files. For example, a bug-fix might require you to edit several files. By submitting these files together as a single change, you record the fact that the editing made in these files corresponds to a single change to the system. The changes have descriptions, and you can enter a text description of what you have changed and why. Changes are numbered so that a newer change always has a higher number.

You can identify a version of a file in 3 ways with Perforce:

1. You can say foo.c#version, where version is a file-specific version number.
2. You can also say foo.c@change, where change is a depot-wide change number. foo.c@115 is a valid identifier even if foo.c was only changed at change 110 and 120. In this case, foo.c@115 refers to the version of foo.c that was changed in change 110, i.e., the version of foo.c that was valid the moment change 115 was submitted.

3. As with many other systems Perforce has labels. Thus, the third way of identifying a version is foo.c@label.

Ease of Use

The normal work with Perforce is done by the following commands:

1. **p4 add** informs Perforce that a local file in your work area is to be added to the files in the depot. **p4 add** only registers your intention to add the file, the actual adding is done by the command **p4 submit**.
2. **p4 edit** is used to open an existing file for editing—the p4d server is informed that you are going to edit the file. On many other systems this process is called checking-out. Changes are actually made by the **p4 submit** command.
3. **p4 submit** does the actual work for the previous two commands.
4. **p4 get** is used to get a file from the depot to the local workspace.
5. **p4 revert**

In addition to the normal commands for editing, there is a fair number of reporting commands such as **p4 describe** to learn more about a change, and **p4 filelog** to list the history of a file. All of these commands are easy and intuitive to use. There is also a **p4 help** command, as well as the man page.

Branches

One of the particularly strong points of Perforce is the way it handles branches, called “inter-file branching”.

On many other systems the branch specification is in some way part of the version numbering or version selection mechanism. This is counter intuitive and is often a cause of confusion. Other systems also make branches of individual files.

Perforce handles branch naming in the same way that you would without a code-management system. In Perforce, the directory my_project/new_branch/ contains the new branch of my_project/old_branch. By making the branch naming a part of the directory tree structure, Perforce has created a very natural way to interact with and think about branches.

In this way a branch is simultaneously created for the complete project, not just for an individual file. This method also helps to keep the branches consistent. A copying algorithm in the server prevents this approach from using more disk space than other approaches.

Above, I have described the normal use of the Perforce branching mechanism. However, the Perforce branching mechanism is even more powerful. It is possible to specify that file trees or individual files are branches of each other. It is even possible to designate two totally unrelated files or directory trees as branches and migrate changes between them.

The specification of branches is done by a branch view. The branch view can contain a simple or arbitrarily complex mapping between file names in the two branches.

Perforce uses the powerful command **p4 integ** and **p4 resolve** to integrate changes between the branches and to resolve conflicts.

Speed

Perforce is a very fast, code-management system. Code-management actions, such as labelling, checking in (**P4 submit**) and checking out (**p4 edit**) are several magnitudes faster than ClearCase, another code-management system.

With Perforce, all normal work such as editing and compiling is done on local files in your work area, making Perforce much faster than most other systems.

Advanced Use

Perforce has a number of advanced features. I cannot list all of them (much less describe them all) in this space, but I will mention a few.

Perforce can have distributed depots. You can run Perforce over WANs, and you can even run it encrypted over the Internet. You can use Perforce with IP-tunneling and firewalls. Perforce can have change submission triggers for external processes.

Perforce has support for off-line clients. That is, it is possible to disconnect a client computer and make changes to the local files in the workspace, and afterwards let Perforce detect the changes and bring them into the depot.

Licensing

You can download all of Perforce, except the license file, from ftp://ftp.perforce.com/. The license file determines how many users the server accepts. Without a license file, there can be only 2 users. The cost of purchasing Perforce (i.e., the license file) is $500US/user. If you purchase Perforce, they e-mail the license file to you. Perforce has announced:

> Non-commercial users of Free-BSD and LINUX may obtain Perforce servers supporting an unlimited

> number of end users gratis. This includes upgrades, but not support. Execution of a Perforce non-commercial license agreement is required.

Answering a direct question, Christopher Seiwald of Perforce, said:

> We cannot guarantee non-commercial users support, but we try not to discriminate between commercial users, evaluation users and non-commercial users.

Perforce is available for Linux x86 and Linux Alpha, as well as for Free-BSD and many commercial systems.

Support

Perforce has provided excellent support to me. No matter how absurd the question or how absurd the task I am attempting, I have always received a good answer by e-mail within 18 hours. Perforce has remarkably good e-mail support. I have asked other customers of Perforce on the Net, and they have all been pleased.

Conclusions

I have used Perforce both professionally (see “MYDATA's Industrial Robots”, *LJ* Issue 39, July 1997) and as a home-hobby programmer. I find Perforce to be a good product with good support. I find other CM products that I have used professionally to be remarkably poor by comparison.

Perforce is not just a technically good product—it is also easy and intuitive to use. Considering the favorable licensing policy of Perforce, I recommend you download Perforce and test it for yourself.

Tom Björkholm is a software engineer at Ericsson Business Networks. He has no connection with Perforce, except as a user. He has used Linux since version 0.95. When not programming, he enjoys sailing. He welcomes comments sent to tom.bjorkholm@swipnet.se.

Archive Index Issue Table of Contents

Advanced search

VAR Station II

James T. Dennis

Issue #44, December 1997

The VAR Station II is obviously intended to be a graphical workstation—I couldn't do it justice if I stuck with my usual suite of text-mode applications.

- Manufacturer: VA Research
- E-mail: sales@varesearch.com
- URL: http://www.varesearch.com/
- Price: $3650 US
- Reviewer: James T. Dennis

A couple of months ago I agreed to do a review of VA Research's VAR Station II. Although I've built or purchased dozens of systems over the years, it's pretty hard to sell me a new system for personal use. When I picked up the review system, I realized that currently my main system was the old 386 I built eight years ago. It's the one I use for reading all of my mail and news, surfing the Web, compiling and testing new software and most of my programming and writing. The beauty of Linux is in how well it supports my old system. With the new system, I knew I was in for a bit of trauma, as it meant forcing myself to use the system for my daily work after almost a decade with my old reliable. (Quit laughing, it's not that funny—well, maybe it is.)

The other thing I knew would be difficult to overcome was my text-mode bigotry. The VAR Station II is obviously intended to be a graphical workstation—I couldn't do it justice if I stuck with my usual suite of text-mode applications. Normally, I've only made the occasional foray into X to use Netscape, ghostview, xdvior, xv and/or xpaint.

Hardware

The VAR Station II is built around a 266MHz Intel Pentium II. This is currently the fastest production x86 processor—although I think I spotted an announcement

for a 300 MHz model on the horizon. Although this may seem tame next to the DEC Alphas at 500MHz (and some as high as 600), there are some advantages to sticking with an x86 processor. The most obvious advantage is that you can install a copy of DOS, Windows 95 or NT if you have occasional need for a Microsoft application, or you're doing any sort of cross-platform development or testing on any other PC OS. Some users also might need access to iBCS (SCO or other PC Unix binary compatibility) or WABI (Windows 3.x under X Windows) or to some other application that's available for Intel Linux but not for any of the ports to other processors (we're thinking of commercial Linux applications here—which are appearing in increasing numbers).

I'll admit I've considered adopting some other hardware architecture—Alpha or PowerPC, maybe even SPARC. However, my informal survey suggests that the various ports of Linux to other platforms have simply not progressed far enough to offer a performance advantage. For whatever reasons, a 500 MHz Alpha just doesn't appear to be 90% faster than a 266MHz Pentium under the current ports of Linux.

Despite the advantages of the x86 architecture for this sort of system, VA Research's choice of the Intel Pentium II chip was also a distressing feature for me. Just prior to picking up the review system I discovered that a new floating-point math bug had been reported (and confirmed by Intel) in that particular microprocessor series. I asked VA Research about this bug. They let me know that Intel would be able to ship a software fix that would patch the microcode in the processor.

The current Pentium II and MMX processors are basically RISC chips at their core—and they run a set of microcode routines that provide the CISC functions. In the case of the Pentium II and MMX chips there is also a small bank of flash ROM where some upgrades to the processor's microcode can be loaded to permanently “fix” them.

At this time, I haven't received the fix disk. However, despite my concerns about the processor, the hardware seemed to perform flawlessly.

Setting Up

Setting up the hardware consisted of unpacking two boxes (monitor and CPU) and plugging in about five cords (monitor, power, keyboard, mouse and Ethernet). With 64MB of RAM and a fast SymBIOS SCSI controller this machine makes my Pentium 150 (another system that I use just for testing and mostly for NT) feel like sludge. I won't even compare this to “old reliable”. The system came with a sound card installed and pre-configured—but no speakers or patch cord. Naturally, the first speakers I acquired came with the wrong sort of cords.

One quirk I noticed about this SCSI adapter—it doesn't seem to support the new bootable CDs. The recent Red Hat releases are written on bootable CDs, which is nice for installation and downright handy for recovering from your latest typo at the root prompt.

Documentation

The VAR Station II ships with a three-ring binder that holds some custom documentation and has vinyl floppy pocket and pamphlet pouches. The first page of the custom documentation is a printout describing how to log in, start X and configure the networking parameters (via the Red Hat Control Panel or a text editor).

They ship the systems with a custom user account already added, so I was able to just log in as “jimd” and go. That is a nice touch and hopefully discourages new Linux users from doing normal work as root.

The other pages list all of the settings for all of the installed adapters including the IRQs, DMA channels and I/O ports that are in use. Although I didn't add any additional adapters, this is the exact information that I hate to track down when I do system upgrades so I'm glad they provide it in such a clean, organized fashion.

Getting Connected

Connecting the system to my household LAN was simply a matter of plugging it in and changing a few lines in the /etc/sysconfig/network file. (Yes, I used the text-editor method rather than the GUI). It only took a few more minutes to add read-only NFS exports between the VAR Station II and “antares” (the old 386). I was then able to access the Internet using IP Masquerading through antares, which dynamically brings up my PPP link using **diald**. There really was no fussing with the hardware—ever.

I wish I could say the same for the software. I think it's a shame to ship a machine that's so clearly intended for use as a multimedia graphics workstation and have it boot to a text-mode login. Remember this statement is coming from an affirmed text-mode bigot. I would ship these systems with a run level of 5 as the default setting, thus providing the new user with an **xdm** (graphical) login prompt.

Video

I also consider it a bit silly to use a 4MB Matrox video card with a depth of only 8 bits. I was able to override this with the command:

```
xinit -bpp 16
```

I also found that Red Baron (Red Hat's preferred web browser) refuses to run in “truecolor” mode. From what I've read, it's possible to use **Xnest** to solve this problem by running an additional instance of the X server on a different display such as :1 and opening a remote client window on it. I had no problems running Xnest, but I couldn't convince it to run with the lower color depth despite several readings of the man page. There were some other programs that didn't like the “truecolor” mode as well.

Despite all of this, there are big advantages to running with the larger color palette if your video card and X server support it. When using **xv** (the X-viewer package) for photo finishing and other graphics packages, you need the extra colors.

X Windows

There were two other nuisances about the way X Windows was set up (which I think was the Metro-X server—though XFree86 is also installed). I don't use X Windows much, as I don't like to wait for it to load. Normally, I start a copy of it and leave it running all the time. When my wife wishes to use the system, I prefer for her to run her own copy of X and leave mine alone. I've always been able to manage this by running my copy on a different virtual display with a command like:

```
startx :1
```

The included version of the **startx** shell script was doing something with a file in /tmp and complained when I tried to start a second session. I worked around that problem by using:

```
xinit xterm :1
```

and manually starting my window manager from the ensuing xterm. Eventually I'll probably fix that startx script.

Also, the SVGA libraries weren't configured at all. No playing **sdoom** on this system without a little tweaking.

After configuring the network and starting X, I was greeted with a couple of xterms and a copy of the Red Baron web browser displaying a page from the localhost web server. This had some information and links to the VA Research and Red Hat web sites (along with a warning that you need to have your Internet connection configured before those would work). However, I would like to have seen more extensive local web pages. What I'd really like to see is a multi-media tutorial and tour—similar to the “Welcome” application that comes

with a Macintosh Performa. This could be done as a set of web pages, Java programs, a Tcl/Tk script or some combination of those options.

The default window manager for recent versions of Red Hat is FVWM 95 which, as the name suggests, is the familiar FVWM configured and tweaked to provide a feel similar to a certain Microsoft product. This is nice for users who are migrating from that universe and reasonably unobtrusive to those of us who are used to FVWM. You can still access the menus by clicking anywhere in the root window (what Windows calls the “desktop” or “background”), and the other mouse buttons still bring up the “Window Ops” and “Task List” menus. FVWM 95 doesn't put any icons (like “My Computer” or “Recycle Bin”) on the root window, so the similarity to the Windows 95 desktop is minimal. However, it is nice that I can use the alt+tab key binding to cycle among tasks in FVWM 95. I'm told you can set up any of the common X Window managers with that feature, but I've never taken the time to edit my rc files to include it.

While exploring the FVWM 95 menus I found familiar problems. So far this has been true of every installation of Red Hat and Slackware I've done. Several of the menu items are non-functional “placeholders” or examples (such as remote xterms to systems that don't exist on my LAN). Also, some of them try to call programs, such as the **xvier** game, that simply aren't installed on the system. Other menu items call their binaries with arguments that generate error messages. There was a new problem, for me, with the “screen lock” and “screen blank” submenus. There are so many blank/lock modules listed on the menu that they push off the edge of the screen—and virtual screen scrolling doesn't work in this context. So there are more screen blankers than you can access. Also some of the screen blanker selections just didn't work (due to typos in the rc file). I consider this to be a fairly minor problem that is easily solved by editing your own copy of the .fvwmrc file—if you understand its syntax.

I'd like to see an rc builder added to the default menus (maybe built around “The Dotfile Generator” by Jesper Pedersen *Linux Journal* Issue 42, October 1997). There's a lot I don't like about the MS Windows and Macintosh user interfaces, but they do have the virtue of being approachable. They have menu items/icons for almost everything and their menu items don't “silently” fail, leaving you wondering what was supposed to happen.

If you don't like FVWM 95, you'd better know how to configure whatever window manager you do like. This installation had a few other window managers installed (twm, olvwm and the old fvwm). However, none of the others had menus that were even close to the system's configuration.

Java

Another problem I bumped into was with the Java demos. There was a tantalizing tree of Java class files under /usr/lib/java/demos. The only problem is that they are applets rather than applications (meaning that they can't be run outside of a browser or viewer). Since neither HotJava nor Netscape was pre-installed and Red Baron and Grail (the two web GUI web browsers that were available) don't support Java, this left a bit of a problem. Here's another case where a few pages on the localhost web server's document tree would greatly help a new user. In this case, I had filed the **appletviewer** command, and so was able to preview all of the demos with the single xterm/bash command:

```
cd /usr/lib/java/demos && for i in */examp*.html;\
do appletviewer $i; done
```

DOSEMU

In contrast, DOSEMU is installed and configured. Maybe I'll get around to digging out my old collection of DOS shareware to play with. For test purposes all I did was run **dos**, change the CONFIG.SYS within the hdimage, restart **dos** and change directories to the mount point for my C drive back on antares, where I was able to run 4DOS and Norton Commander with no problems. For the record, that was accessing a DOS file system mount on another Linux system through an NFS mount on the local host. It worked without any special fussing.

My only complaint about the way that DOSEMU is configured on this system has to do with the permissions. It started out as SUID root (which is necessary for technical reasons) and world executable. A more conservative and reasonable approach is to create a dos group—and change the binary's permissions to group executable—stripping all access from “other”. This suggestion applies across the board to almost all SUID programs. It limits the number of users who can attempt to exploit security problems with the files. If you routinely add all of your real users to dos and other groups, at least you prevent possible access through pseudo users like bin, www and nobody.

Operating System

Like other systems that I've had with the OS pre-installed, this one had a “cookie cutter” feel to it. For example there are two partitions: / and /home. The consensus of Unix system administrators is that it's safer to make a relatively small root partition, a medium or large /usr, an alternate root and some other partitions suited to a particular host's needs. A separate /var/spool partition—or even /var/spool/mail and /var/spool/news—are often a good idea. I like to create a large partition to use for /usr/local (on some systems I make /home a symlink to /usr/local/home).

There are many other things I do when setting up systems for myself or my customers—like turning on the immutable bit on system binaries and share libraries (see the Linux-Tips HOWTO) and removing extraneous services from inetd.conf (the single most effective thing you can do to improve your system's security). While I don't expect VA Research or Red Hat to do these by default, it would be nice to see some of these practices codified as install options.

Software Installation

After exploring as much of the pre-installed software as I could find, I installed some other packages. I started with a copy of Mathematica and a copy of Applixware. Those installed easily and ran fine. I fussed with GNUStep DR2 (which is the free NeXT/Open Step clone that's currently being developed), but I never did get it to run. Another package that gave me grief was Lyx (WYSIWYG LaTex word processor).

That is about par for the course. About a third of the programs that I tried to build from sources, and maybe a sixth of the RPM packages I experimented with, didn't build or install on the first try. This is better than my average with Windows programs, and worse then my experience with DOS shareware (although not by much).

For comparison, I recently purchased a Mac Performa which I've been setting up to send to my mother. I bought a couple of CDs full of Mac shareware -- mostly games—and played with them to decide which ones to install for her. I found that over 90% of the games and tools could be run right off the CD with no problems—and no installation. For those Powertools that I did decide to install, each was simply a matter of dragging the folder from the CD into my file window (group). In fairness I have to point out that one of the programs that did not behave under MacOS 7.5 managed to completely trash the system drive when I tried it again under the new MacOS 8.0. Norton Utilities was no help for the damaged file system (so now I'm reinstalling everything from scratch). The failure rate was much lower—but more catastrophic.

Now this may seem like an unfair comparison. Linux isn't “supposed” to be as easy to use as MacOS. However, the point I want to make is that Linux is catching up quickly. I wouldn't send my mother a Linux system this year—but in another year or two, I might.

Red Hat 4.2

After working with the system as shipped for several weeks I broke down and installed a fresh Red Hat 4.2 onto an external hard drive that I added to the VAR Station II. Red Hat shipped 4.2 shortly after I picked up this review system. However, I didn't upgrade it right away because I wanted to do a fair review of

the system as it was shipped. This serves two purposes. One, it tests how well-supported this mix of hardware is by “stock” Linux distributions (and reveals if any custom or special drivers or kernel patches are required). Two, it probably matches the version of Linux that would be installed if you were to get one of these systems now. I picked up a copy of the Red Hat 6 CD set which includes the free portions of Red Hat (no Metro-X server) and mirrors of the sunsite and ftp.x.org sites.

The SymBIOS SCSI controller in this system has a SCSI-3 connector, but VA Research thoughtfully provided a SCSI-2 adapter. Adding the extra drive was simply a matter of picking an unused SCSI ID, and plugging in a cable and terminator.

I normally do upgrades as new installations on separate drives so I can easily revert to the old, working system whenever something “suddenly” doesn't work on the new installation. This installation went smoothly, with the usual fussing over the lilo.conf to convince it that I really did want two bootable partitions on two separate drives. The new twist this time was that I needed an **initrd** directive.

This new version is much nicer. More of the initially installed menu items work out of the box (at least if you do an install of “everything”), and there are some nice new packages (like the **lincity** game—a “civilization” for Linux). By far my favorite new package is XEmacs.

XEmacs is an improved Emacs (based on the GNU version—and therefore free). As the name implies XEmacs has enhanced support for fonts and graphics when run under X Windows. The “GNUscape Navigator” (formerly called w3-mode)--a web browser package written in the Emacs lisp macro language—can render pages with embedded graphics under XEmacs but is limited to text mode (like Lynx) under GNU Emacs. The surprise for me is XEmacs' support for **ncurses**, which allows me to have color and “fontlock” support for my text-mode screens and from my laptop when I log in over a serial line.

Conclusion

This is an excellent combination of hardware for Linux. It is probably the fastest single-processor x86-based system available (VA Research has some multi-processor server systems, too), and all of the equipment works with Linux and is pre-configured for it. If you're tired of fighting with your hardware to get your sound card to play CDs or your video card to work in decent resolutions, you should get a system like this one. The only things missing from the hardware package are the speakers and some sort of suitable backup and bulk storage device.

While I have my concerns about Intel's processors, I am sure that they will fix the problem. The Linux community will work around them if Intel takes too long. I wouldn't build any bridges, aircraft or medical equipment without running the numbers through the same calculations on another architecture, but I've recommended that since the first FDIV bug was announced, and I recommend it regardless of which processor(s) are involved.

Finally, the software configuration is still rough in spots. Red Hat is showing steady improvement, and VA Research does offer other distributions (such as Caldera and Craftworks), if you ask.

VA Research clearly tries to balance the degree to which they customize their installation against the expectations and preferences of experienced Linux users. They also have to do constant research as hardware vendors make unannounced and undocumented changes to various components (which might cause an Ethernet card of a given model to suddenly stop working with Linux, for example), and as the development of Linux and other software that runs under it keeps steaming along. Personally I think they've done an excellent job—there are very few PC manufacturers and integrators that are willing to take up the challenge.

So, if you're tired of getting blank stares when asking vendors about Linux and you need a fast X Windows workstation, get a VAR Station II from VA Research.

Jim Dennis is the proprietor of Starshine Technical Services (http://www.starshine.org/). His professional experience includes work in technical support, quality assurance and information services for both large and small software companies. He has just begun collaborating on the 2nd edition of a book on Unix systems administration. Jim is also an avid science fiction fan. He can be reached via e-mail at info@mail.starshine.org.

Archive Index Issue Table of Contents

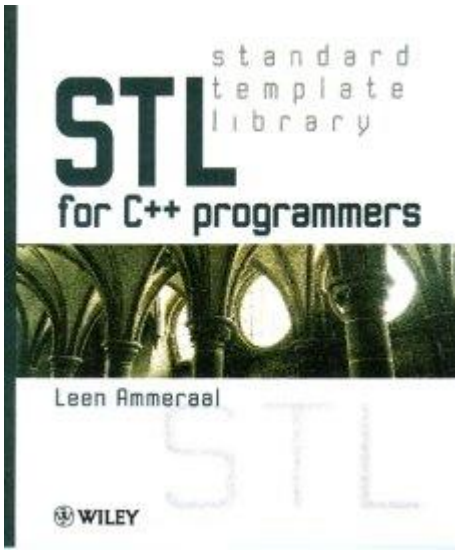
Advanced search

STL for C++ Programmers

Bob Adkins

Issue #44, December 1997

Although the primary focus is not Linux, his many excellent examples are easily adapted to the g++ environment in Linux 2.0++.



- Author: Leen Ammeraal
- Publisher: John Wiley
- Price: $49.99
- ISBN: 0-471-97181-2
- Reviewer: Bob Adkins

Good news—STL, Standard Template Library, is alive and well on Linux. Leen Ammeraal demonstrates this as well as his considerable skills as a master teacher in *STL for C++ Programmers*. Although the primary focus is not Linux, his many excellent examples are easily adapted to the g++ environment in Linux 2.0++ (his examples adapted for Linux can be found at http://www.cwareco.com/download.html. Ammeraal writes concise yet thorough explanations on each aspect of using STL.

What is STL?

STL started in the 1970s with Alexander Stepanov's ideas about designing general algorithms. Stepanov, together with Meng Lee, took these ideas to HP and developed the first C++ based STL. By 1994, STL was accepted into the C++ draft standard by the ANSI/ISO C++ standards committee.

STL distinguishes general algorithms from the more specialized data and methods encapsulated by ordinary abstract data types. In this way, complex and powerful algorithms can be implemented independently of the data to which they are applied, allowing for generalization and reuse of these algorithms. In STL more familiar object abstraction is reserved for data and methods. These are then tailored and bound to the characteristics of their underlying container type such as sequence containers and associative containers. Examples of this distinction with respect to sequence containers, such as vector objects, are begin, end and insert. These methods access and manipulate the underlying data of the vector container class. However, these methods are specific to the treatment of the data and should not be confused with more general algorithms such as **find**, **sort** and other advanced numeric algorithms (e.g., accumulate or inner product). General algorithms are a kind of method abstraction in contrast to more traditional data abstraction.

STL on Linux: What's broken. What's not.

STL support has been available on Linux since GNU's libg++ 2.6.2. Now with release 2.7.2.1, the library is quite usable for most major features with the exception of name space scoping. There are minor differences with other implementations, such as Borland's BC5 environment, but these differences mostly concern header naming conventions. There is also a curious problem with **fstream** which involves an unexpected file access mode default.

g++ 2.8.0 will offer a more complete STL based on newer code from SGI and a complete redesign of the compiler's template implementation. Unfortunately, g++ 2.8.0 is not expected to fix the problems with using name spaces.

Overview

From the beginning Leen Ammeraal presents a quick and practical startup for the STL beginner. He then explains how to use the sequence containers (vectors, lists and deques), the associative containers (sets and maps) and, later, examines containers derived from these basic types such as stacks, queues and priority queues. As a simple application, he shows how to to build a telephone directory using associative map containers. Later, he demonstrates a more complex map application, called a concordance, which produces a line-oriented index of all words in a text file. He also shows function objects which can be

used to build custom ordering relationships among the elements of a container. He moves on to algorithms and the practical details of STL's generic algorithms for manipulating sequences and for sorting. He demonstrates the built-in numeric algorithms which make STL attractive for implementing statistical analysis such as the Least Squares Method.

How Big Can You Count?

As his final chapter, Ammeraal presents a wonderfully fun example of “Very Large Numbers”. Here Ammeraal uses STL to calculate pi to an arbitrarily large number of digits. Ammeraal exploits the power of STL to reduce the implementation complexities of defining and operating on extremely large numbers. He notes that, thanks to the STL's vector container, this version is “simpler and more elegant” than an earlier solution he presented in his book *Algorithms and Data Structures in C++*.

For added spice, I modified his program to generate a histogram of the digits computed for pi. At 100,000 places, digit “1” is a very slight favorite. Moreover, with this example Linux shows its strength. After turning on full g++ optimization, I was able to calculate these 100,000 digits in just under 20 minutes. Under DOS/Windows, Ammeraal indicated that this same calculation took several hours using BC5.

Bob Adkins is a software engineer and CEO of cWare, Inc., a software development company specializing in Web content distribution using database technologies under Linux and other Unix platforms. For the past several years Bob has moved proprietary systems to more open, flexible and collaborative software solutions. Bob enjoys traveling to Europe and India where he finds that Linux together with Web information and commerce applications have the potential to level the international playing field for rich and poor countries alike. He can be reached via e-mail at cwareco@erols.com.

Archive Index Issue Table of Contents

Advanced search

Linux Multimedia Guide

Michael J. Hammel

Issue #44, December 1997

Jeff's book covers a wide range of material related to the creation and use of multimedia files with respect to the Linux operating system.

- Author: Jeff Tranter
- Publisher: O'Reilly & Associates, Inc.
- E-mail: nuts@ora.com
- URL: http://www.ora.com/
- Price: $32.95 US
- ISBN: 1-56592-219-0
- Reviewer: Michael J. Hammel

I picked up my copy of *The Linux Multimedia Guide* by Jeff Tranter back in December of last year, the same month it hit the shelves of my local computer book store. I wrote a fairly supportive review of the text in my “Graphics Muse” column in the *Linux Gazette*, right after having put to bed my multimedia chapter for the *Unix Web Server Book*. At the time I thought I understood multimedia fairly well, since the research for my book had been fairly thorough.

When *Linux Journal* contacted me about doing a review of this text, I jumped at the chance. I figured I could just spiff up my old review and pass it along. It took some time to get around to doing this “spiffing”, and in the intervening period, I realized that this wasn't fair. I needed to go back and re-evaluate the review and the text. Linux users deserve up-to-date information. Besides, I had learned quite a bit more about multimedia.

As it turns out, most of my initial inspection was right on the money. Jeff's book covers a wide range of material related to the creation and use of multimedia files with respect to the Linux operating system. The text is approximately 350 pages, including source code listings for a number of sample multimedia

applications. As usual, O'Reilly provides copies of the source from their ftp site. Jeff's coverage of multimedia is thorough, although it lacks detail in a few areas and provides information on subjects that may not be necessary for most multimedia-application developers.

When I first discovered this book, I thought “Rats, Jeff beat me to it.” Much of what Jeff covers is listed in my own Linux Graphics mini-HOWTO (LGH); however, a number of items, such as audio, are not covered by the LGH. There is more detail about video formats and tools and detailed programming considerations for various hardware (CD-ROMs, joysticks and sound devices). All of these make the *Linux Multimedia Guide* a good addition to the O'Reilly family of Unix books. The text is decidedly aimed at developers and administrators. The actual use, from an end-user perspective, of the various applications is not covered in any serious detail by this text.

The *Linux Multimedia Guide* is divided into five sections:

1. Introduction to Multimedia
2. User's Guide
3. A Survey of Multimedia Applications
4. Multimedia Programmer's Guide
5. Appendices

The first section introduces the reader to the various concepts involved with multimedia such as the CD-ROMs, image file formats and sound files. The chapters are generally brief, and the one on audio is quite informative. Jeff is the originator and maintainer of the Linux Sound HOWTO, and his experience shows in his coverage of audio issues. There is a discussion on audio file formats as well as a comparison of a few popular sound cards available for Linux.

Section two opens with a discussion on hardware requirements for doing multimedia on Linux systems. Most of this section centers on either the CD-ROM driver or the Linux sound driver, known at various times as Voxware, TASD and OSS Lite and now available commercially from 4Front Technologies (http://www.4front-tech.com/). There is also a short chapter on the joystick driver.

The second longest section, “A Survey of Multimedia Applications”, covers applications for the various forms of multimedia. There are chapters on sound and music applications, graphics and animation applications, hypermedia applications and games. When I first reviewed this text, I hadn't really considered games as real multimedia applications, mostly because I seldom play games on my computer. I now realize that was a mistake—one need only

look at the big computer retail chains' shelves of MS and Mac games to realize this fact. Games implemented as network applications using Java, JavaScript and the new Tcl/Tk plug-in for Netscape are not covered, but general X Windows, SVGA and command-line style games are discussed briefly.

The third section is very similar to the LGH in that the chapters provide the program names and URLs associated with them (if any). The number of applications covered is less than in the LGH, but there are better descriptions of the applications in the book; plus Jeff covers games and audio applications that the LGH does not.

Chapter fourteen opens the fourth section, the “Multimedia Programmer's Guide”. This section is the longest in the book and covers all the devices discussed earlier. Other chapters in this section cover some of the available graphics and windowing toolkits available to multimedia developers. One chapter contains three sample applications.

I didn't say much about this section in my first review but I need to do so now. First, the chapter on programming sound devices is quite good and the sample programs easy to follow. If you're going to program sound for your applications, you definitely should read this chapter.

The chapters on programming the joystick and CD-ROM devices are out of date (the joystick chapter) or irrelevant (CD-ROM) for current multimedia applications. I say this because the future of multimedia on Linux is through X Windows, not SVGA. Joysticks and other pointing devices should be programmed through the X Input Extension (which XFree86 3.3 now supports and the commercial X Server vendors Metro-X and Xi Graphics are rumored to be adding soon) and not via direct management of the device. To be fair, Jeff's book did come out before this support was publicly released by XFree86, and it is only recently that developers have started to make wide use of this server extension.

A CD-ROM's association with multimedia applications is nothing more than another file system, accessed using the usual file system APIs. In general, multimedia-application developers do not need to know how to write drivers for the CD drives. Device driver information is not generally provided in texts on multimedia for other platforms (except for unusual devices). Again, to be fair, I'm not certain whether information on how to write drivers for new CD drives is provided in other texts. This chapter's appearance in the Linux Multimedia Guide is not a detriment to the text, but it is extraneous to multimedia-application developers.

Chapter 17's discussion on graphics and windowing toolkits provides some good reference material but not much detail. It would have been nice to see Jeff take a stance on which toolkits might be most beneficial for developers to start with or at least examine more closely. Also, Jeff missed any reference to OpenGL. Of all the toolkits he mentions, only Java is likely to be of greater interest than OpenGL to Linux multimedia developers. OpenGL and Java are widely accepted standards (real or de facto) whereas most of the other toolkits mentioned are, at best, simply multi-platform. OpenGL is available from both commercial X server vendors (Xi Graphics and Metro-X) using software acceleration only (with hardware acceleration planned) and is also available via Brian Paul's terrific Mesa/GL package.

In general, I find the *Linux Multimedia Guide* a good reference text with a moderate degree of developer tutorials. Unlike many of the books available for Linux, this text provides a detailed explanation of the various programming interfaces; it is a useful tool beyond the simple “what is this and where do I get it” that many of the HOWTOS provide. The main drawback that I see is that, like most other Linux texts, this text does not provide a user's perspective on any of the tools listed. If Linux is ever to go beyond a developers-only platform, there will need to be detailed user's guides for well-known applications. Still, Jeff does a good job covering the extent of a large topic. Although not likely to be of much interest to experienced developers, I have no problem recommending this book for the beginner or the intermediate user or developer.

Michael J. Hammel (mjhammel@csn.net) a Computer Science graduate of Texas Tech University, is a software developer specializing in X/Motif living in Denver, Colorado. Michael writes the monthly Graphics Muse column in *Linux Gazette*, maintains the Linux Graphics mini-HOWTO, helps administer the Internet Ray Tracing Competition (http://irtc.org/) and coauthored *The Unix Web Server Book*, published by Ventana Press. His outside interests include running, basketball, Thai food, gardening and dogs.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Industrializing Web Page Construction

Pieter Hintjens

Issue #44, December 1997

This article describes a Perl HTML preprocessor that takes the work out of building web pages.

When I started building my company's web site about a year ago, I looked for a good, visual web editor, and finding one quickly produced some nice web pages. A week later, I had thrown the web editor away and was working on a tool to solve some of the major difficulties I had found. In this article I'll look at the result—a free HTML preprocessor written in Perl—that makes mass production of web pages a feasible and economical task.

htmlpp was one of the first Perl programs I wrote, and I've not regretted the choice of language. Perl allows me to add functions to the program as fast as I can think of them. The consequence is that htmlpp is a very rich tool, making the task of maintaining a web site with thousands of pages easy.

There are at least a dozen free HTML preprocessors available today; I know of three with the name **htmlpp**. Something is driving people to write these programs, but what? Some 95% of the web pages I produce are on-line documentation, and I dislike building these by hand. Each page needs a standard header, footer and appearance. When I change my mind, it takes a lot of mouse clicks to go through each web page again, and a lot of care to make sure that every page conforms to my preferred style.

Thus, I started htmlpp with the idea: “take a large text file and break it into smaller web pages, adding pretty headers and footers, building the table of contents, cross-references and hyperlinks.” It would also be nice to define symbols like **$(version)** and place them into the text. How about conditional blocks so that I can generate frame and non-frame web pages from the same document, a way to share definitions between projects, a **for** loop to build structured text, access to environment variables and Perl macros, some more hot coffee and a raisin bagel?

htmlpp uses the term “document” to refer to the text files it inputs. This is a “hello world” document:

```
   .echo Hello, World.
```

Here's something more involved:

```
    .define new-year 0101
   .if "&date("mm-dd")" eq "$(new-year)"
   .  echo Happy New Year!
   .else
   .  echo Hello, World.
   .endif
```

If you've used C or C++, htmlpp looks very much like the C preprocessor. You get commands like **.define**, **.include** and **.if** that work in a similiar fashion to the C preprocessor equivalents. For instance, the .if command works at “compile time”, i.e., when you build the HTML pages, not when they are displayed by the browser. Some other htmlpp commands were borrowed from the Unix shells.

Note how I define a symbol, **new-year**, and then use it in the document as **$(new-year)**. **htmlpp** provides many variations on this theme; for example, the $(*...) form creates a hyperlink:

```
   .define lj http://www.ssc.com/lj/
   $(*lj="Linux Journal"<\n>) is the magazine of the Linux community.
```

To define a counter which runs from 0 upwards:

```
    .define counter++ 0
```

A realistic htmlpp script uses the .page command to create HTML pages. Listing 11 shows the template file supplied by htmlpp for your new projects.

New Project Template

Each HTML page gets a header and a footer. **htmlpp** lets you construct very complex headers and footers. This footer, taken from the htmlpp documentation, builds hyperlinks to the first, previous, next and last pages in the document, plus an index that lets the user jump to any page in the document.

```
    .block footer
   <HR><P>
   | $(*FIRST_PAGE=<<) | $(*PREV_PAGE=<)
   | $(*NEXT_PAGE=>) | $(*LAST_PAGE=>>)
   .build index
   <P><A HREF="/index.htm">
   <IMG SRC="im0096c.gif" WIDTH=96 HEIGHT=36 ALT="iMatix"></A>
   Designed by <.HREF "/html/pieter.htm" "Pieter Hintjens">
   © 1997 iMatix
   </BODY></HTML>
   .endblock
```

The **.build index** command builds the index by making a list of all the pages in the document. With an .if command, we can show the current page in relationship to the other pages. This is how I define the index:

```
.block index_open
<BR>
.block index_entry
.if "$(INDEX_PAGE)" eq "$(PAGE)"
| <.EM $(INDEX_TITLE)>
.else
| $(*INDEX_PAGE="$(INDEX_TITLE)")
.endif
.endblock
```

This code is beginning to get a bit complex, but the results are well worth the effort. The symbols in capital letters (e.g., **$(PAGE)**, the file name for the current HTML page) are supplied by htmlpp. Some of these symbols, such as **$(NEXT_PAGE)**, require that htmlpp go over the document several times. In fact, htmlpp will run through the document three or more times, until all cross references have been resolved. This multi-pass approach can be a little slow, but it is powerful enough to handle the footer block shown above.

Figure 1. Screen Shot

The **.build toc** command builds a table of contents, a vital part of any large document. **htmlpp** comes with a small file, contents.def, that does this job. To build the table of contents, you do the following:

```
.include contents.def
```

The contents.def file first defines three blocks (toc_open, toc_entry and toc_close) and then does a **.build toc**:

```
 .block toc_open
<MENU>
.block toc_entry
<LI><A HREF="$(TOC_HREF)">$(TOC_TITLE)</A></LI>
.block toc_close
</MENU>
.end
<P>
.build toc
<HR>
```

htmlpp uses such predefined blocks for headers, footers, indexes, table of contents and other constructions. You can define your own blocks in order to

pull standard chunks of HTML text into your pages. You can also use **.include** commands, but this practice can lead to the creation of many small files.

The key to unlocking htmlpp's real power is learning a little Perl. When you use the .if command, for instance, you use Perl. So, I can write something like this:

```
    .if $ENV {"RELEASE"} eq "test"
```

It's also possible to run Perl programs and pipe the output into your HTML pages or to extend htmlpp's syntax with your own functions. Finally, since htmlpp comes with source code under the GNU General Purpose License, you can change the tool in any way you wish.

At the other extreme, you can use htmlpp in “guru mode” to turn a simple text file into structured HTML pages. All you need to do is mark the section headers. **htmlpp** inserts a table of contents, breaks the document into pages, adds headers and footers, detects numbered and bulleted lists, paragraphs, tables and so on. This is a quick and lazy way to produce useful HTML pages without tagging every paragraph.

To use htmlpp, you have to be happy writing HTML by hand (unless you work in guru mode). In return, you get an economical way to maintain large web sites without losing any control over the quality of your work.

To install and use htmlpp, you need Perl version 4 or 5. Download htmlpp from http://www.imatix.com/ and unpack the .zip file. The package comes with HTML pages describing how to install and use. If you have questions, comments or suggestions, don't hesitate to send me e-mail.

Pieter Hintjens is a programmer and the founder of iMatix, an Internet software company. You can download the latest version of htmlpp, and find-out more about the free software that iMatix produces, from their website at http://www.imatix.com/. He can be reached via e-mail at ph@imatix.com.

Archive Index Issue Table of Contents

Advanced search

CGI Developer's Resource

Reuven M. Lerner

Issue #44, December 1997

Overall, I thought that the book was fairly well-organized, although a number of sections were of questionable value.

- Author: J. M. Ivler with Kamran Husain
- Publisher: Prentice Hall, Inc.
- URL: http://www.prenhall.com/
- Price: $49.99 US
- ISBN: 0-13727-751-2
- Reviewer: Reuven M. Lerner

"CGI Developer's Resource" is one of many books released in the last year on the Common Gateway Interface. CGI enables HTTP servers to return not just the contents of an HTML file, but also the HTML-formatted text output from a program. (Like many other books on the subject, this book appears to have been rushed out.) There is a lack of attention to detail and the example programs are not well structured and are poorly presented.

Overall, I thought that the book was fairly well-organized, although a number of sections were of questionable value. Books cannot possibly explain all things to all people, and while descriptions of client-server programming, of thin vs. fat clients and of HTTP were all worthwhile, I felt that an explanation of how different types of firewalls work was probably unnecessary.

I disliked the authors' programming style, which used very few subroutines. If the authors were writing short programs, the lack of subroutines would seem reasonable, but there were several five-page program listings that lacked even a single subroutine. (In addition, there were too many unbroken blocks of code in the book.) True, the programs contain extensive comments—some of them holdovers from when the code was discussed and improved on the Internet, and others useful explanations of what the code is doing—but I would have preferred to see them designed as well as they were documented. Given that the front cover claims that the book demonstrates good software design, it would have been nice to see more modular code, perhaps broken up into reusable libraries.

The authors purposely ignored the existing CGI libraries for Perl, preferring to process CGI-related data on their own. Moreover, the authors explicitly decided against using Perl 5 (in favor of Perl 4), because of its object-oriented features; because it “would have required the explanation of the use of modules and OOP in Perl” and because “the folks supporting some of the Perl code shown here are not Perl programmers.” Even non-Perl programmers can and should learn about Perl's error-checking and security mechanisms, which trap problems and make CGI programs safer and easier to debug.

Indeed, very little space was dedicated to the tricky problem of debugging CGI programs or of how to construct error messages that make it easy to find problems in CGI code.

One of the chief benefits of the CGI standard is its portability across platforms. However, as experienced CGI programmers know, portability is guaranteed for the standard itself, not for the languages or operating systems in which CGI programs are written. I appreciated the author's note on the first page of the preface, in which they acknowledge that “this text is a bit Unix-centric”—but that understates the non-portable nature of most of the code in the book. True, every program can be modified so that it will run on other platforms, and they require far fewer modifications than would be necessary if the programs were written in C. However, to claim that the programs are portable is a bit far-fetched, given their reliance on external Unix utilities.

A number of editing and production issues also bothered me: a large number of typographical errors and misspellings, a poor choice of font in program

listings, explanatory notes printed on a dark background that makes some words almost illegible and an index that is far too small. There was also no mention of DBM files or relational databases, two data-storage technologies that every CGI programmer will probably use at some time.

There were several nice parts to the book. The program for a monthly-activities calendar was quite good, especially since it allowed for two different views of the same data. There is an extensive treatment of server-side includes, which contained a listing of variables specific to SSIs. Also, a mention of how to write CGI programs that retrieve data left by system utilities in flat files is useful information. The authors explicitly named the requirements for each project before embarking on it—something that I wish more software engineers would do in today's bug-infested world.

Overall, $50US seems a bit steep for this book, given that the programs are often similar to those you can find on-line, the explanations contain bugs and inaccuracies and the code is poorly organized. With some serious editing, this book could have been quite interesting—but in its current incarnation, the book is too weak to stand on its own.

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

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Keeping Programs Trim with CGI_Lite

Reuven M. Lerner

Issue #44, December 1997

This article is an introduction to a second CGI module, CGI_Lite, that is not as feature rich as CGI but is much smaller in size and therefore more efficient for use in small-scale projects.

This month, we will look at CGI_Lite, a Perl 5 module written by Shishir Gundavaram. CGI_Lite is one of several modules available for CGI programmers; the best known of the bunch is CGI.pm, written by Lincoln Stein. Indeed, I have used CGI.pm in nearly every “At the Forge” column, as well as in many programs over the last few years, on many web sites.

While CGI.pm is useful and rich in features, it is also large, weighing in at a hefty 153KB. On my Red Hat 4.2 system with 40MB of RAM, starting Perl 5 and loading CGI.pm uses about 2.7 percent of the physical memory—over 1MB—before I have even allocated any data structures. On a popular Web server, it is easy to imagine how many CGI programs running simultaneously would lead to a heavy load, both on the CPU and on the server's memory, leading to a significant slow-down.

There are a number of solutions to this problem, including using a language other than Perl for CGI programs. This month, though, we will look at another solution: CGI_Lite.pm, a module that does less than CGI.pm but is much smaller and faster. CGI_Lite.pm takes a mere 17KB on disk, and when loaded into memory along with Perl 5, takes only 2.0 percent of the physical memory on my system, about 800KB. This is still a relatively large amount of memory, but given that invoking Perl 5 uses about 560KB, it strikes me as a reasonable trade-off.

CGI_Lite.pm is not a panacea; it leaves out a number of useful features that have made their way into CGI.pm over the years. However, if your CGI programs require only a limited set of features and you would like to keep your programs as trim as possible, you might want to consider using CGI_Lite in at least some of your programs.

Getting Started with CGI_Lite.pm

Before you can use CGI_Lite, you need to get a copy from CPAN (the Comprehensive Perl Archive Network), a set of FTP and web servers that make Perl code, documentation and utilities available to the public for free. As of this writing, the latest version of CGI_Lite is 1.8, meaning that you can retrieve it from the URL http://www.perl.com/CPAN/modules/by-module/CGI/CGI_Lite-1.8.tar.gz.

If CGI_Lite has been updated by the time you read this, you might need to change the numbers in the last part of the URL. Once you have retrieved the module, you can unpack it with the command:

```
tar -zxvf CGI_Lite-1.8.tar.gz
```

which uncompresses (**-z**) verbosely (**-v**) the file (**-f**). This action creates CGI_Lite-1.8 on my system. Then perform the standard Perl module installation as follows:

```
perl Makefile.PL
make
make install
```

Note that you may have to be logged in as root in order to install CGI_Lite on your system.

Once the module is installed, you can use it in any program by including the line:

```
use CGI_Lite;
```

at the top of your program.

Of course, including a module is the easy part—learning how to use it can be a bit more complicated. Let's see how to use CGI_Lite.pm by creating a simple program, one which expects to receive a user's first name from an HTML form. When the form is submitted, the program prints a short personalized greeting to the user. If you are wondering why we are starting with an HTML form and the POST method, rather than the simpler GET method, stay tuned—it is harder than you might think.

Listing 1. HTML Form with Single Text Field

Listing 1 is a simple HTML form, containing a single text field called *firstname*, that we can use for our test. When a user clicks on the submit button in this form, the *firstname* text field is sent via the POST method to the program called

/cgi-bin/hello.pl. Listing 2 shows one way in which we might write **hello.pl** using CGI_Lite.pm.

Listing 2. Initial Version of Perl Program hello.pl

In short, this program executes the following actions:

1. Imports the CGI_Lite module.
2. Creates an instance of CGI_Lite.
3. Retrieves the HTML form elements into a hash, also known as an "associative array".
4. Uses the value of the *firstname* form element to return a string to the user.

Dissecting hello.pl

Now that we have gotten an overall picture of what is happening in the above program, let's look at this in greater detail, with a bit of attention to some of the differences between CGI.pm and CGI_Lite.pm.

In CGI.pm, we can retrieve form elements using the **param** method. When invoked in a scalar context, param allows us to retrieve the value of a single HTML form element. For example, if we have defined **$query** to be an instance of CGI, we can place the value of the *firstname* field in the **$firstname** variable with the following statement:

```
my $firstname = $query->param("firstname");
```

If we invoke param in an array context, then we get a list of all form elements that were submitted to the program. For example, if we want to put the names of all HTML form elements into the array **@names**, we can do so with the following statement:

```
my @names = $query->param;
```

We can then iterate through **@names** to retrieve and print the value associated with each form element, as in:

```
my $element = "";
    foreach $element (@names)
    {
        print "<P>$element = ",
         $query->param($element), "</P>\n";
    }
```

We can accomplish this with CGI_Lite.pm, but in a slightly different way. CGI_Lite.pm has a single method for retrieving form elements, one which uses hashes rather than a mixture of scalars and arrays. To retrieve form elements,

we use the method **parse_form_data**, which returns its results in a hash. Retrieving individual form elements is thus a two-step process. First we put all of the elements into the hash, and then we retrieve the one in which we are interested:

```
my %FORM = parse_form_data;
my $firstname = $FORM{"firstname"};
```

If we want to get a list of the form elements that were sent, we can use the keys function. Thus, to put the names of the form elements in the array **@names**, we can type:

```
my @names = keys %FORM;
```

We can even get them in alphabetical order, by prefacing keys with a call to sort:

```
my @names = sort keys %FORM;
```

We can print the names and values of all form elements by iterating through **@names** and retrieving the values in which we are interested:

```
my $element = "";
foreach $element (@names)
{
print ",<P>$element = ", $FORM{$element},
        "</P>\n";
}
```

Creating Variables from Form Elements

If we know that we want to put one or more of the form elements into scalar variables (and not keep them in the hash), we can do so by calling the method **create_variables**. For instance, in our example above, we first had to use parse_form_data in order to get the form elements into the hash %FORM. Then we had to assign **$firstname** in a separate step. If we had wanted to assign 10 variables based on the contents of the form, we would have needed to make 10 separate assignments, which is rather inefficient.

To get around this problem, we can use the **create_variables** method, which automatically creates local variables for us. If we want to turn each form element into its own variable, we can simply invoke:

```
$query->create_variables(\%FORM);
```

When this method returns, we have a new variable defined for each element that was in the submitted form. Thus, if we have a form element named *firstname*, the value associated with that element is now available via the variable **$firstname**. The backslash in front of %FORM gives us a reference to the hash, a new feature in Perl 5 documented in great detail in the Perl manual pages (available by typing **man perlref** on most Linux systems).

There is one potential problem with create_variables, namely, your program might define variables with the same names as one or more form elements. For example, Listing 3 is a version of hello.pl in which we give the variable **$firstname** a value and call create_variables on the submitted form that included an element named *firstname*.

Listing 3. Second Version of Perl Program hello.pl

When **$firstname** is set to the value NOT CHANGED, as in Listing 3, the value of the HTML form element *firstname* is ignored when we call create_variables, and we get a greeting to NOT CHANGED, rather than the user's first name. If we comment out the line defining **$firstname** as NOT CHANGED, create_variables does its job just fine, creating a variable named **$firstname** and giving it the value that the user provided. This behavior is a good idea in terms of web security, but the silent failure of one or more variable assignments strikes me as a possible pitfall.

CGI.pm offers similar functionality with its **import_names** method. In this case, the authors encourage users to import names into a separate name space, ensuring that there are no name conflicts with existing variables.

Notice that in the Listing 3 version of hello.pl, I have removed the **use strict** line. This was to avoid possible conflicts when commenting out the line that defines a default value for **$firstname**. The **strict** module requires that you define variables before using them; however, if we are referencing variables that are created by create_variables, this is impossible.

GET and POST

CGI_Lite.pm is smart enough to grab form elements passed by either of the two methods: GET or POST. POST is generally considered to be the better method of the two, since it passes the contents of the form to a CGI program via standard input (stdin), rather than as part of the URL. However, if we were interested in passing a name to hello.pl as part of the URL, we could do so as follows:

```
http://localhost/cgi-bin/hello.pl?firstname=Reuven
```

Of course, if you are testing this program from a computer other than the web server, you need to replace **localhost** with the name of a server. For example, if your server runs on www.fictional.edu, you could use:

```
http://www.fictional.edu/cgi-bin/hello.pl?firstname=Reuven
```

Notice how we can set the variable's value after the question mark, known in CGI lingo as the “query string”. The query string is part of the URL, and URLs may not contain white space or other “dangerous” characters that might be misinterpreted by the browser and/or the server. For these reasons, certain characters must be sent in “percent-hex” format, in which the character's ASCII value in hexadecimal is prefaced by a percent sign. Obviously, the percent sign itself (ASCII value 0x25) must be encoded in this way. Thus, if my “first” name were actually two names, I could send the string as follows:

```
http://www.fictional.edu/cgi-bin/hello.pl?firstname=J%20Edgar
```

Since the “space” character is ASCII 0x20 (32 in decimal), we can insert a space into the URL by sending a **%20**. CGI_Lite.pm automatically translates the percent-hex encoding into the ASCII codes we want.

Getting the Query String

While GET can be used to send name,value pairs, it is often used to send simple text strings. For example, it might be nice to send a name without assigning any value, as in:

```
http://www.fictional.edu/cgi-bin/hello.pl?J%20Edgar
```

This technique is often useful when CGI programs have to receive a user's unique ID in a relational database running on the web server. If we send the identifier as part of the query string, the program can grab that value and use it as the index into a table in the database, producing a personalized home page or otherwise unique output customized for the user.

Several on-line booksellers use this method. When I go to Amazon.com to check the status of my latest order, I go to a URL that looks like:

```
https://www.mybookstore.com/cgi-bin/order.pl?1234-5678-9012
```

What I would like is a simple way of retrieving this string. CGI.pm allows you to get the string by pretending that the contents of the query string are assigned to the variable named keywords, so if we are using CGI.pm, we can type:

```
my $id_number = $query->param("keywords");
```

Now, the variable $id_number contains the value “1234-5678-9012”.

If we are using CGI_Lite.pm, things get a bit more complicated, because the module expects the query string to only be used for sending name,value pairs, not individual text strings. Thus, when we send the above query string, CGI_Lite.pm assumes that what we are actually setting the form element

named “1234-5678-9012” to a null value—not quite what we might expect, but something which we can manage.

One possible method is to load parse_form_data to turn the received name,value pairs into a hash. The hash contains a single key, corresponding to the data that was passed in the query string, which CGI_Lite.pm thinks is a variable name. We can then retrieve that key by getting the list of keys in our hash. Listing 4 is code that accomplishes that feat.

Listing 4. Using Hash Keys

This is not the most efficient way to get the information, but it does do the trick. We could simply read the information from the QUERY_STRING environment variable—but that would introduce another problem, namely, the translation of characters sent in percent-hex encoding. By using the built-in facilities of CGI_Lite.pm, we ensure that the translation is done correctly.

If you find this somewhat confusing, you're not alone. Many of my own programs take advantage of the query string, and having to pretend that my data is really a variable name strikes me as a bit odd. Perhaps a future version of CGI_Lite.pm will handle this, although adding too many features would eventually turn it into CGI_NoLongerLite.pm.

Debugging

Debugging CGI programs is often difficult because the execution takes place behind the scenes. In contrast with more typical programs, which allow us to interact with them while they are running, CGI programs are invoked by Web servers, with input coming from the user's Web browser (via the Web server, which hands that data to the program), and with output returning to the user's browser (again, via the Web server).

CGI.pm offers two good aids to debugging CGI programs. A **dump** method that prints out the contents of all HTML form variables as they are received, and a command-line interface that allows programmers to enter variable assignments without invoking the program from an HTML form.

In keeping with its light-weight philosophy, CGI_Lite.pm does not offer the command-line debugging interface, which might make debugging large programs difficult. However, it does offer a way to check the data that was received from the user's web browser. The **print_form_data** method sends all of the known name-value pairs to stdout. If your program does not work correctly and you want to check the values of the input data, you can add the following line to your program:

```
$query->print_form_data;
```

Which Should You Use?

With the above discussion in mind, which module should you use when writing your CGI programs? In most cases, I would tend to stick with CGI.pm, for a variety of reasons.

First of all, I tend to use CGI.pm's command-line debugging interface quite a bit, and the fact that CGI_Lite.pm lacks such an ability is a major hindrance for me. It is certainly possible to get around this problem, since I wrote CGI programs for a while before CGI.pm appeared on the scene, but it never hurts to have another debugging tool in your arsenal, particularly when writing large, complicated programs.

A second reason why I would tend to favor CGI.pm is because I often have to work with other people on projects, and using two different interfaces to the CGI standard might make life difficult for them. (We have enough problems as is; we don't also need to try to remember whether we should be using param or parse_form_data in order to retrieve information.)

Third, I find it useful to have extra functions that take care of the small parts of producing HTML. I used to constantly forget to put two newline (**\n**) characters at the end of MIME headers; with CGI.pm, I no longer have to remember.

At the same time, I find it somewhat irresponsible to write small CGI programs that use over 1MB of RAM before they even begin to perform any calculations or allocate any data structures. For small projects in which I want to use Perl (rather than a compiled language, such as C) but in which I want to maximize efficiency, I use CGI_Lite.pm. I also like the use of hashes, which strikes me as a natural way to store and retrieve form elements. Also, the fact that CGI_Lite.pm does almost everything I wish, including such advanced items as HTTP cookies and the uploading of files, makes it rather attractive for small-scale projects.

In an era of software bloat and programs that try to do more and more, it is refreshing to find a module that tries to do less and does it well. CGI_Lite.pm is not appropriate in all cases, but it is useful, well documented and efficient. If you are trying to squeeze the last few ounces of memory and CPU time from your web server, consider using CGI_Lite.pm in your next program—and enjoy the extra RAM for other projects.

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

Letters to the Editor

Various

Issue #44, December 1997

Readers sound off.

Linux Certification

There is a discussion on the GLUE (Groups of Linux Users Everywhere, http://www.ssc.com/glue/) e-mail list about a certification program for Linux. Most of the discussion is positive. Most writers feel that a comprehensive certification will make great strides in enabling Linux to be used in the business sector with more confidence. I would like to know how a larger audience feels about it. For any certification program to be representative, it must be a cooperative effort of all the major Linux companies. A relatively subjective organization would need to head the testing such as the LDP or SSC.

—Bryan Coleman, Triad Linux Users Group bcoleman@triadsearch.com

Web browser

I thought Phil Hughes's May article on web browsers for Linux was a little negative. [“Linux and Web Browsers”, Issue 37] Here is some more positive news: Sun's HotJava 1.0 browser is available from http://java.sun.com/products/hotjava/. When you look at the distribution formats it has versions for Windows and Solaris. However, the browser is written in Java, just called from a different shell script for the two platforms. Download the Solaris version and run it under JDK 1.1.1 to get a working browser for Linux.

Sun MicroSystems, Inc. has bundled the HotJava web browser with the Java Runtime Library for SunOS on SPARC hardware. The download file is now 8.5MB.

Another choice is the Plume browser (formerly Surfit!) by Steve Ball at the Australian National University (http://tcltk.anu.edu.au/). This runs under Tcl/Tk 8.0. It is still under development, but Steve is actively working on it. What's

more, you get the source code so you can do things with it too. The current version of Plume is v0.62alpha.

—Jan Newmarch jan@ise.canberra.edu.au

Linux and Ham Radio

As an avid amateur radio operator and a Linux tinkerer for nearly a year, I'd like to say a hearty “Thank You” for the positive coverage you give my favorite hobby in *Linux Journal* [“Packet Radio Under Linux”, Jeff Tranter, September 1997] and *Linux Gazette* [Issues 10 and 11]. Of course, it's great having access to the only OS that supports the packet radio protocol. Most other big-time magazines wouldn't bother to print such articles, but it proves the editorial commitment you have to covering all relevant aspects of Linux: business, technical, hobby/recreation and more.

I'm always glad to see this type of article as it introduces ham radio to a larger audience. We're always looking for more hams willing to push the digital RF (Radio Frequency) envelope. I invite all interested parties to get their license and join in building a state-of-the-art, wireless, non-commercial TCP/IP network.

—Nate Bargmann KA0RNY nfbargma@notes.up.com

Big Brother

If I'm not mistaken, anyone on the Internet can execute any command on a machine with the CGI scripts you published on page 58 of *LJ*'s August issue [“Big Brother Network Monitoring System”, Paul M. Sittler].The script executes **$TRACEROUTE<\!s>$***, so a cracker can feed it with a machine name such as www.tamu.edu; then type **cat/etc/passwd** to see the last command being executed.

In my opinion, CGI scripts should all be written in Perl with the **-T** option set (**-T** tests that the file type is text, not binary) and should include the line **use<\! s>strict**. Strict compliance for symbolic references, global variables and key words—violations cause immediate program abend. The Bourne shell is especially dangerous. At least, enclose the arguments between double quotes.

—bortzmeyer@pasteur.fr

Security Blanket

I don't know that I have ever written to a magazine editor before, but Lee Brotzman's contribution to the August 1997 *Linux Journal*, “Wrap a Security Blanket Around Your Computer,” was very timely and very well written.

One of my client's Linux systems came under the control of hackers (who fortunately were somewhat benign in their apparent intentions for this particular system) about the time I received the aforementioned copy of *LJ*. During an intense weekend of observation and examination of various system logs, I was able to determine how the system had been compromised. After considering various strategies (and reading the issue cover to cover), I used Mr. Brotzman's article as a cookbook to install a series of TCP wrappers while continuing to watch the hacker's activities.

Not really knowing the expertise of the hackers, I surmised they were also "cook-booking" and decided to slowly cut off their air supply, in order to see what alternative methods of access, or back doors, they may have established. Selectively applying TCP_wrappers enabled me to do just that, and I received quite an education in the process. Today, thanks to *LJ* and Lee Brotzman, my client's system is secure, and I have greatly increased my understanding of security from an administrative perspective.

Once again, thanks for publishing useful, accurate information. If you have an award for writer of the year, I would like to nominate Lee Brotzman for his clear, concise presentation of an important topic. If you don't, start one.

—Mel Lester meljr@connet80.com

Various Comments

I would like to commend you on such a fine publication. It is good to see a magazine published on a regular basis about Linux, that has such excellent content. I would just like to make a few comments.

My first comment is regarding a couple of letters to the Editor from the September issue (Issue 41). The two letters are "More Novice Articles" and "More Technical Articles". I feel that *LJ* has a nice balance of both novice and advanced articles, which makes it useful for the beginner, the guru and everyone in between.

My second comment is about the article "Robocar: Unmanned Ground Robotics" by Kerry Kruempelstaedter (also Issue 41). I found this article very interesting, although I believe that many of the pictures were missing from the article. The first figure is Figure 5, and 1 through 4 are...nowhere in site.

And finally, I have a question regarding your article titled "Quota: Managing Your Disk Space" (and again, issue 41). Towards the end of the article January says that to give everybody on your system the same quota use **edquota -p <prototype> ***. If this gives everybody on your system the same quota.

Wouldn't this cause some problems? It is fine that all users get the same quota, but what about root, bin, daemon and other such users?

Overall though, it was yet another excellent article.

—Philip Cox phil@yucc.yorku.ca

Yes, pictures were left out of the Robocar article. There was not room for all of them, and the ones in the magazine did not get renumbered.

It's true that you would not want to set root, etc. to the same quota as users. In the article, Mr. Rooijackers mentions setting soft and hard to 0 for those accounts that should not have a quota. In the quota HOWTO, it recommends turning quota on and off for various partitions. Thus, if all your users are on one partition, while root is on another, you can use the above command to set the quota for all users, while not setting it for root.

London Times?

I don't know why you invented a non-existent paper to blame the Hewson anti-Linux article for in *LJ* #40 [From the Editor]. You get it right in the footnote (*The Sunday Times*), but there is no such paper as *The London Times* as in your heading. *The Times*, the paper some people think published the article, is a different newspaper from *The Sunday Times*.

Congratulations on the generally high standard of *LJ*.

—Keith Briggs, England kmb28@cam.ac.uk

Sorry about that—the reference to it I had said the London Times. The editor who checked the article later and added the footnote should also have changed the reference in the body of the article.

Microstation95 for Linux

The article on Microstation for Linux, in the July issue was nicely done and well written. The only problem is in the section “Installed Base” where it states “The review kit states that there are over 200,000 users of Microstation...” I am sure they don't include academic versions in that number. If this is the case, then none of the 200,000 are running Linux. In addition, the academic version is all that is available for Linux and to get it you must prove you are a student or faculty member of a four-year-degree university.

The sad truth of the matter is that Bentley, and for that matter most other software companies, don't get enough requests for Linux ports of their

product, to justify the production costs. To illustrate this point, here is a portion of an e-mail I received from Phil Chouinard, a Bentley Representative, on August 4, 1997.

> ... to date, a complete commercial port hasn't happened. About the only thing that is holding us back right now is a commitment from the business community. Note that we do not get involved or wrapped up in OS wars—there is no winner in such (in fact, we've heard the same arguments from MacOS loyalists, OS/2 followers, etc.) Since we have commercial products running on these OSes where others don't, and we have also committed to the Internet—recently licensing Java from Sun, convincing us to do a full -fledged port really won't convince the business community to go to Linux, but the opposite may be true.

The above, I fear, is typical of the software community as a whole. Linux is now at an awkward moment. If business will use Linux, software companies will write programs for it. If software companies will write programs for Linux, business will use Linux. Neither wants to be first. How do we get out of this catch 22?

—Dave Blond elln9jdz@ix.netcom.com

More information on Bentley Systems, Inc., publisher of MicroStation 95, can be obtained from their web site at http://www.bentley.com/academic/products/order.html.

Programming with the Xforms Library

After typing in the source code for xhello.c (July 1997, page 71), I tried to compile it using the author's suggested command line. Unfortunately for me it didn't work. I received a very vexing error from /usr/lib/libforms.so complaining of undefined functions. After fighting it a bit more, I decided to look at forms.ps and found a difference between the command line suggested in the manual and the author's. To make a long story short, this command line worked on my system:

```
gcc -o xhello xhello.c -lforms
        -lX11 -lm
```

It has all the same elements as the author's, only in a different order. I am running a Slackware '96 system with kernel version 2.0.30, Xfree86 3.3 and xforms 0.86. I guess I'll go with what works.

—Nate Bargmann nfbargma@notes.up.com

Linux and NDS

First off, I would like to congratulate you on a job well done. It is a credit to the Linux community to have such a well-written and timely publication devoted to it.

I was wondering if any development has been done on integrating NDS (Novell Directory Services) into Linux? Novell has recently announced the release of the NDS source code to developers. Sun, HP and SCO have taken the hint and announced that each will provide NDS support in upcoming releases of their OSs. I have yet to find a place where Linux does not have the potential to replace a commercial operating system, and I think that support of a commonly used directory service would give a network manager even one more reason to use Linux instead of SCO ODT, Solaris, HP-UX or Netware.

—Tim McQueen mcquetm@mail.auburn.edu

Caldera is in the process of porting both NDS and NetWare File and Print services to Linux. Both these technologies currently require Streams, so Caldera is building the necessary infrastructure into Linux to port these technologies. For more information, please e-mail Caldera at info@caldera.com.

New Chip Technologies

I have been unable to follow current chip discussions on the Net on the basis of these two new technologies: AMD's K6-233MHz with MMX processors and Intel Pentium II, 233&266MHz with MMX processors

Are these technologies compatible with Linux? Is MMX programming likely to be taken advantage of in Linux?

—Michael T. McGurty mmgurty@ovis.net

In answer to your first question, yes. Intel and AMD design backward compatability into all new processor chips to assure software compatability.

As to the second, Linux programming is likely to use advanced MMX technology sometime in the future, but I have no answer as to when—hopefully, this year. Of course, compatablity with the installed user base is a concern. It would have to be a kernel option. Multimedia extensions (MMX) optimize graphical video display (X Windows) and sound subsystems performance.

Archive Index Issue Table of Contents

Advanced search

Promoting Linux

Marjorie Richardson

Issue #44, December 1997

If business will use Linux, software companies will write programs for it. If software companies will write programs for Linux, business will use Linux. Neither wants to be first. How do we get out of this Catch 22?

This month's Letters to the Editor includes an e-mail about the product review of Bentley Systems' MicroStation95 [July 1997] that I found quite interesting. In particular, in the last paragraph the writer, Dave Blondell, says:

> Linux is now at an awkward moment. If business will use Linux, software companies will write programs for it. If software companies will write programs for Linux, business will use Linux. Neither wants to be first. How do we get out of this Catch 22?

That is exactly the question that we all must answer if Linux is to continue to expand in use and become a major player in the OS game.

While there is probably more than one answer to this question, one of the best and easiest is marketing. When we, as at-home Linux users, promote Linux in our workplace as an efficient, reliable and inexpensive alternative to the other available operating systems, we take the first step in solving the conundrum posed above.

Each month *LJ* includes a “Linux Means Business” article for the express purpose of letting the world know the many ways in which Linux is being used in the workplace. Bentley Systems might be interested in the fact that Linux is being used to design items as diverse as clothing [“Using Linux at Lectra-Systèmes”, Pierre Ficheux, April 1997] and integrated circuit boards [“An Introduction to IC Design Under Linux”, Toby Schaffer and Alan W. Glaser, July 1997]. These articles and others [e.g., “IMEC/NIT”, Erwin Glassee and Rudi Cartuyvels, December 1996] show that there are companies doing CAD work

using Linux that would provide a market for products such as Bentley's MicroStation95.

Last month, Martin Sjolin provided us with a great way to promote Linux at work [“Linux Expo at UBS”, November 1997]. Mr. Sjolin along with other Linux users put on an Expo at the Union Bank of Switzerland where they are employed. His description of the steps they took to put on the Expo serves as a blueprint for others to follow.

In June of this year, the Atlanta Linux Enthusiasts (http://www.ale.com/) joined with Linux International to put on a successful, national trade show devoted exclusively to Linux: The Atlanta Linux Showcase. An article describing how your group can organize a successful show like A.L.S. appeared in Issue 19 of *Linux Gazette* (http://www.ssc.com/lj/issue19/trade.html): “User's Groups and Trade Shows, Lessons from the Atlanta Linux Showcase” by Andrew Newton. This paper can also be found on the G.L.U.E. (Groups of Linux Users Everywhere) web site at http://www.ssc.com/glue/. This example makes the point that users' groups and Linux Advocacy teams are always a good resource for inventive ideas for promotion—join one today.

Kelper's Day Parade

On Sunday August 31, *Linux Journal* helped spread the word about Linux by sponsoring a float in the annual Kelper's Day Parade in Pacific Beach, Washington. Now, I have no idea what a Kelper is—a gatherer of kelp perhaps?—but I had a lot of fun participating in the parade. My favorite float was a front-loader with a dummy in the scoop and a sign that read: “Call 911. We Scoop and Run.”

LJ's contribution was a huge *Linux Journal* banner followed by Phil's old blue Mercedes with a computer fastened to the top. A bunch of us employees walked (and rode on the front fenders) the two-mile parade route giving out Linux stickers, magazines and candy. (The kids liked the candy the best.) Phil took pictures and, after the parade, fed us barbecued salmon and beer—he's a nice guy, after all. Next year he has offered to feed all the Linux geeks that show up.

Linux Journal CDs

And now, for a little self-promotion—*Linux Journal* has put together a CD containing a browser and all of the 1996 issues in HTML format. This CD is a nice, convenient way to keep all that great Linux information at your fingertips in an easy-to-search format. It also takes up a lot less space on your bookshelf than 12 printed magazines. Get yours today.

Linux Journal Index Archive

In our past December issues, we included an index of all the articles that have appeared in *Linux Journal* from our inception. This index has gotten too long to print in its entirety, so we have placed it on our web site, http://www.linuxjournal.com/.

Archive Index Issue Table of Contents

Advanced search

A Confession and Some Ramblings

Phil Hughes

Issue #44, December 1997

Where is “Stop the Presses”? Well, that's the confession.

Where is “Stop the Presses”? Well, that's the confession. When we begin layout of an issue of *LJ*, we reserve a page for “Stop the Presses”. Although we usually have some ideas, the news for STP has to happen in that one week between our regular deadline and the STP deadline—this month it just didn't happen. Unlike the supermarket tabloids, we decided to confess instead of just making something up and printing it. Hence this editorial.

Last month I talked about Linux going mainstream. This morning I received my ballot to vote for the Board of Directors of UniForum, a Unix trade association. There are five members on the Board and ten candidates. Two of those candidates are Linux people: Jon Hall and Bob Young. I voted for both. Having two members on the board is another good way to get Linux integrated into the greater Unix community.

As I write this, the Linux community is preparing for the Comdex show in Las Vegas. Our own Carlie Fairchild is in charge of working with Linux International members and the Linux community as a whole to have a serious Linux presence at the show. At this point the Linux pavilion includes twelve vendors. With an expected attendance of 250,000, this show will definitely be a great opportunity to get the word out about Linux.

On to Systems Administration

The focus of this issue is Systems Administration. In the traditional Unix community, systems administration is generally done by an elite group of individuals. This isn't necessarily bad as the average user shouldn't have to be concerned with hard drives, processor interrupts and Ethernet cables. After all, computers are just tools.

There are two reasons that systems administration is more integrated into the domain of the average user in the Linux community. First, many of the community members are running stand-alone computers with PPP connections to the Internet or work in small companies where there is no full-time systems administration talent.

The second reason is that many Linux systems are customized to fit the requirements of a particular company or even embedded into a product. For example, Schlumberger is using Linux in a point-of-sale system [“Highway POS System” by Marc L. Allen, November 1997]. The user of this system is a convenience store clerk, but the people who have worked on developing the applications software are also involved in the operating system.

The result of this integration is that there are more people working on systems administration tools and issues with Linux than in traditional Unix environments. Thus, more issues are raised, and more tools are created.

If you look at the Usenet newsgroups that cover Linux you will continue to see posts discussing configuration issues, selection of the best installation methods and other systems administration issues. This open discussion is what has made distributions like Caldera, Craftworks, Debian and Red Hat evolve. The developers get a chance to see real systems administration issues raised and then create solutions they consider to be the best way of dealing with them.

The features in this issue address setting up Linux as a proxy server, user administration, dealing with pluggable authentication modules and systems information retrieval. Other articles focus on RAID disks, tracking system disk usage and an article on teaching systems administration. Whether you are that person at home with a Linux system or a systems administrator at a huge multi-national corporation, there should be something here to interest you—you may even learn something new.

Jump in, take a look and then let us know what you want to see next year, as we intend to offer an annual systems administration issue.

Archive Index Issue Table of Contents

Advanced search

Linux in Camouflage

Lieutenant Colonel Hart

Issue #44, December 1997

Here is how the U.S. military is using Linux to save you money.

As the Army's demand for real-time, battle-command information increased, computer systems became more complex, expensive and unwieldy. To counter the proliferation of single-use systems, the Army's Battle Command Laboratory (http://cacfs.army.mil/) at Fort Leavenworth entered into a rapid prototyping agreement with Mystech Associates to design a new Maneuver Control System for the Army.

Maneuver Control System/Phoenix (MCS/P) was approved by the Department of the Army in November 1995 to be issued to Army units for training and feedback. While not the final answer, it is the best system available for providing interoperation between the Army's “command and control” computer systems. In January 1996, MCS/P was used successfully by the 4th Infantry Division and III Corps in a Battle Command Training Program Warfighter exercise. MCS/P is currently in use in Bosnia and other theaters around the world.

MCS/P and its sister intelligence system, ASAS Warlord Notebook, share the same underlying code and provide the military with the applications shown in the “Applications” box.

Applications

Originally developed to run on Sun SPARC 20s, MCS/P and Warlord Notebook have been successfully ported to Linux systems by Mystech. Porting these applications to Linux provided a cost-effective solution to employing these systems on a large scale. The tactical version (hardened and ruggardized) of the SPARC 20 costs the U.S. taxpayer approximately $48,000.00 per machine; the non-tactical or commercial system about $18,000.00. The commercial Pentium 166 machines running the Linux port cost approximately $3,000.

Figure 1

The National Guard Bureau realized the benefits and importance of training its commanders and their battle staffs to operate effectively and efficiently in this information-rich environment. The Leader Development Center (LDC) was given the task of developing and implementing procedures to infuse these systems and technologies into the National Guard. In September 1996, the LDC established a training site at its Fort Leavenworth facility and issued Linux systems to the National Guard's 34th and 38th Divisions. These two divisions will use Phoenix and Warlord Notebook during their 1997 Battle Command Training Program Warfighter exercises at the LDC.

Phoenix and Warlord Notebook provide National Guard commanders at multiple echelons with enhanced situational awareness and a relevant common picture of the battlefield overlaid on digital maps. These two systems provide the commander near-real-time combat information over battlefield communications from their battle staffs, subordinate commanders and intelligence assets.

System Operation

Systems

The software systems are comprised of government owned, GNU licensed and other freely distributed software. The current X Window System is XFree86. To a Linux purist the file structure is wrong. Since the official platform to run these applications is the tactical SPARC, the file structure mirrors that of the SPARC. While confusing at first, this change eliminates the problems caused by operating on multiple systems with different file-system structures and simplifies file transfers.

Each computer has both Phoenix and Warlord Notebook installed, and is its own mail host and web server. The user ID of the individual using the system determines which application system is started.

Military operations are map and graphic intensive. Instead of paper maps, digital map products from the National Imagery and Mapping Agency (http://www.nima.mil/) are used. To reduce the bandwidth required for communication, full digital maps reside on each system. The only map or graphic material actually transmitted are the geo-referenced graphics. Geo-referencing allows the re-scaling of maps and graphics without loss of clarity or the danger of graphics being out of position. (See Figure 2)

Map with Maneuver Graphics

Information on friendly and enemy unit activities and locations is loaded into the system at the first point of observation or occurrence and is then sent throughout the network. Once entered, the data does not have to be re-entered anywhere along the network. The user is unaware that the majority of information is sent using normal e-mail. He clicks on the operation that he needs to be accomplished, e.g., send updated unit locations, and the system queries the appropriate data bases, prepares the message and sends it.

Operations orders are centered around the maps and maneuver graphics. Now, instead of taking the graphics, converting them to words, transmitting the words which are then converted back into graphics on the receiving end, the information is sent as graphics with a minimum of words. An ingenious use of HTML and CGI provides the capability to build complex operations orders with full graphics and map references. Then, with CGI, these orders are available to be either pushed or pulled throughout the network.

A collaborative white board is built in. This provides the commander with a rapid means to clarify his points or make last-minute changes.

One of the most perplexing problems in data entry on tactical systems used to be that the user had to know a myriad of archaic formats and had to enter those formats as text. Phoenix and Warlord Notebook uses simple fill-in-the-blank forms and lets the computer figure out where the “/” and “.” characters should be placed. (See Figure 3 and Listing 1.)

Fill-in the Blank Message

Listing 1

Training

The Leader Development Center's computer training facility is a networked suite of twelve student systems and one instructor system. A 42-inch Mitsubishi monitor allows the students to easily follow the instructor.

LDC Classes

The classes are kept small, with two instructors for twelve students. The students who attend training at the LDC are the key staff and trainers for their division. Our low student-faculty ratio ensures that the students master the concepts and skills required to successfully use these systems with their home units. The student success rate has been extremely high. Some of those who have attended are currently using Linux on their home or business systems or, are systems administrators on Unix or other systems.

Using equipment and software provided by the Leader Development Center, each division has established its own training facility and program. The experience from these division training programs is providing the basis for fine tuning the LDC program for implementation in the other National Guard Divisions.

Future

The use of Linux has accelerated the development and issuance of computer systems to the National Guard and U.S. Army. The decreased cost of hardware and software actually makes the systems affordable. Without Linux, these complex and important systems would not be available.

Joel D. Hart started playing with computers in 1970 programming in PL/1 using 029 card punches. He earned a B.S. and a M.S. at Clemson University and a Ph.D. at Mississippi State. Even after being badgered by his son, he still hasn't switched to Linux on his home machine. He can be reached via e-mail at hartj1@leav-emh.army.mil.

Archive Index Issue Table of Contents

Advanced search

New Products

Amy Kukuk

Issue #44, December 1997

Samba 1.9.17, Laptop Accelerated-X Display Server, MetaCard 2.1.1 and the MetaCard Starter Kit and more.

Samba 1.9.17

The Samba Team has announced version 1.9.17 of Samba, a suite of corporate network integration tools. Samba is designed to service any Server Message Block (SMB) client. The new release features CIFS support, increased speed, a larger variety of servers, larger networks and improved browsing. For more information and downloading, go to or ftp://samba.anu.edu.au/pub/samba/MIRRORS.txt. Samba is freely available under the GNU public license.

Contact: URL: http://samba.canberra.edu.au/samba/samba.html.

Laptop Accelerated-X Display Server

Xi Graphics has released its X server for laptops. The Laptop Accelerated-X Display Server enables laptop users to utilize workstation-like features such as multiple visuals and gamma correction. Like the desktop version, this product supports PseudoColor, High Color and True Color modes and handles large displays. For more information see Xi Graphics' web site.

Contact: Xi Graphics, 1801 Broadway, Suite 1710, Denver, CO 80202, Phone: 303-298-7478, Fax: 303-298-1406, E-mail: info@xig.com, URL: http://www.xig.com/.

MetaCard 2.1.1 and the MetaCard Starter Kit

MetaCard 2.1.1 and the MetaCard Starter Kit, a GUI development and multimedia authoring tool, are now available from the MetaCard WWW and FTP sites: http://www.metacard.com/ and ftp://ftp.metacard.com/MetaCard/. New features include support for HTTP, POST, GET and proxies, encoding functions

and new scripting language features. A full list of features is available in the README.2.1.1 file.

Contact: MetaCard Corporation, 4710 Shoup Pl., Boulder, CO 80303, Phone: 303-447-3976, Fax: 303-499-9855, E-mail: www@metacard.com, URL: http://www.metacard.com/.

pryzm

Rondeau Software announced the release of pryzm, a Motif-based text editor for Linux with CDE. pryzm includes file opening via “drag and drop” from the CDE File Manager, session manager compliance, navigation to any open file in any CDE workspace and more. The price for a single-user license wh binaries is $50US.

Contact: Rondeau Software, Inc., 24 Dawn Heath Drive, Littleton, CO 80127, Phone: 303-904-3604, E-mail: rondeau@csn.net, URL: http://www2.csn.net/rondeau/.

QuickStart

Enhanced Software Technologies, Inc. (EST) has announced the availability of QuickStart, a new family of multi-platform software solutions for system replication and disaster recovery. The family consists of QuickStart System Replicator, QuickStart System Rescue and QuickStart Data Rescue. QuickStart's system replication and disaster recovery capabilities are ideal for open systems VARs and Integrators. QuickStart System Replicator licenses are available for 10 to 10,000 users, starting at $25US per user. QuickStart Data Rescue is priced at $25US per user. The QuickStart WARStart Package is $499US.

Contact: Enhanced Software Technologies, Inc, 5016 S. Ash Ave., Suite 109, Tempe, AZ 85282, Phone: 602-820-0042, FAX: 602-491-0865, E-mail: lj-info@estinc.com, URL: http://www.estinc.com/.

TIMESERIES by Empress

The Empress RDBMS has added an option for selecting the best indexing method for a specific application. In addition to the B-tree indexing method, users of the Empress RDBMS can now take advantage of a new index method called TIMESERIES. Visit their web site or send e-mail to sales@empress.com.

Contact: Empress Software, 6401 Golden Triangle Drive, Greenbelt, MD 20770, Phone: 301-220-1919, E-mail: sales@empress.com, URL: http://www.empress.com/.

TalentSoft Web+ 3.0

TalentSoft announced TalentSoft Web+ 3.0, a web-based application development tool for Unix (including Linux) and Windows. The key features of Web+ are support for mixed platforms and a multi-tiered client/server architecture. New features include more speed, improved handling of variables and the ability to call functions from DLL files. Evaluation copies are available at the TalentSoft web site.

Contact: TalentSoft, 900 Nicollet Mall, Suite 700, Minneapolis, MN 55402, Phone: 612-338-8900, Fax: 612-904-0010, E-mail: info@talentsoft.com, URL: http://www.talentsoft.com/.

Open Sound System Sound Drivers for Linux

4Front Technologies announced the availability of the Open Sound System sound drivers for Linux (OSS/LinuxPPC) running on CHRP-based PowerPC workstations. OSS/LinuxPPC is a commercial implementation of the OSS/Free drivers, the soundcard drivers provided with the Linux kernel distribution. It supports built-in audio devices found in PowerPCs as well as Plug-n-Play (PnP) soundcards. A 7-day trial version of OSS/LinuxPPC is available at 4Front Technologies' web site.

Contact: 4Front Technologies, 11698 Montana Avenue, Suite 12, Los Angeles, CA 90049, Phone: 310-820-7365, Fax: 310 826-2465, E-mail: info@4front-tech.com, URL: http://www.4front-tech.com/linuxppc.html.

GNU Emacs Version 20

The Free Software Foundation has released a new version of its GNU text editor and programming environment. New features include support of international character sets, single multibyte character encoding within an Emacs buffer, coding systems translation and file locking with NFS. The source code for GNU Emacs 20 is available for free download from GNU ftp sites.

Contact: Free Software Foundation, URL: http://www.gnu.ai.mit.edu/.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The Linux RAID-1, 4, 5 Code

Gadi Oxman

Ingo Molnar

Miguel de Icaza

Issue #44, December 1997

A description of the implementation of the RAID-1, RAID-4 and RAID-5 personalities of the MD device driver in the Linux kernel, providing users with high performance and reliable, secondary-storage capability using software.

Using RAID (Redundant Array of Inexpensive Disks) is a popular way of improving system I/O performance and reliability. There are different levels of disk arrays that cover the whole range of possibilities for improving system I/O performance and increased reliability.

This report describes the current implementation of the RAID driver in the kernel, as well as the changes we made to the kernel to support new disk-array configurations that provide higher reliability.

The Multiple Devices (MD) Driver

The MD driver is used to group together a collection of block devices into a single, larger block device. Usually, a set of SCSI and IDE devices are configured into a single MD device. As found in the Linux 2.0 kernel, it is designed to re-map sector/device tuples into new sector/devices tuples in two different modes (personalities): linear (append mode) and striping (RAID-0 mode).

Linear mode is just a way of concatenating the contents of two smaller block devices into a larger device. This can be used to join together several small disks to create a larger disk. The size of the new disk is the sum of the smaller ones. For example, suppose we have two disks with 300 sectors each; after we configure them as linear MD devices, we have a new MD device that has 600 sectors: the sectors 0 to 299 of the device are mapped to the first disk and the sectors 300 to 599 are mapped to the second disk.

RAID-0 mode (also known as striping) is more interesting. This mode of operation writes the information to the device while distributing the information over the disks that are part of the disk array. Unlike linear mode, this is not just a concatenation of the disk-array components; striping balances the I/O load among the disks resulting in a high throughput. This is the personality chosen by most people who want speed.

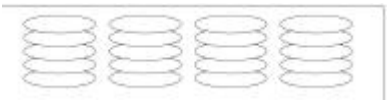


Figure 1 shows how four disks are arranged in this mode. Shadowed regions are those that provide redundant information, and those stacked-up disks represent a single disk. As you can see there are no shadowed regions in the figure. What does this mean? Well, it means that if there is a hardware problem in any of the elements of the disk array, you lose all of your information.

Both the linear and the striping personalities lack any redundancy and error recovery modes. If any of the elements of the disk array fail, the contents of the complete MD device are useless, and there is little hope that any useful information can be recovered. This is similar to what happens with regular secondary storage devices—if it fails, you lose your information. However, with RAID-0, you have a higher risk of losing your information than with a regular disk. The higher failure rate is due to the fact that you have more disks and a failure in any of the disks make the RAID-0 contents unusable.

If you have a good backup strategy and you don't mind losing a day of work if any of your disks fail, using RAID-0 may be the best thing to do. For example, RAID-0 is used for newsgroups like comp.unix, but a higher reliability RAID level is used for important newsgroups like alt.binaries.pictures.furniture.

The way these two personalities are supported by the MD driver in the kernel is quite simple; the low level **ll_rw_blk** routine is responsible for putting block driver I/O requests on the system-request queue. This routine must be modified to call a mapping function that is part of the MD driver and is invoked whenever a request is issued for a block on a MD device.

The ll_rw_blk routine conceptually looks like this:

```
ll_rw_blk (blocks)
{
    sanity-checks ();
    for-each block in blocks {
        make_request (block);
    }
}
```

It is modified to support the striping (RAID-0) and linear personalities in this way:

```
ll_rw_blk (blocks)
{
    sanity-checks ();
    for-each block in blocks {
        if (block is-in md-device)
            md_map (block)
    }
    for-each block in blocks {
        make_request (block);
    }
}
```

Block re-mapping happens just before the input/output request is put into the system-request queue. This re-mapping function is quite simple. It is invoked with pointers to the device and to the block number, and all it does is change the device ID and the block number. The device ID is changed to point to one of the disks in the disk array, and the block number is changed to point to the proper location on that disk. Basically it is a nice hack (but it uses a couple of “ifdefs”, which we all know our fearless leader does not like).

The Extensions for RAID-1, 4, 5

The MD mapping function works fine for re-mapping personalities such as the linear and the striping personalities. Unfortunately, it is not enough to support the RAID-1, RAID-4 and RAID-5 personalities. The reason for this is that these modes cannot do their job by just re-mapping a given device/sector pair into new device/sector pairs; all of these new personalities require complex input/output operations to fulfill a single request.

For example, the mirroring personality (RAID-1) duplicates the information onto all of the disks in the array, so that the information on each disk is identical. The size of the disk array is the size of the smallest disk (the lowest common denominator). Therefore, it is not a good idea to make a RAID-1 with a 100MB disk and a 1GB disk, since the driver will just provide 100MB on each (and ignore the fact that you have 900MB free on the second disk).

The MD driver, when asked to write information to the device, queues a write request for all of the devices in the array with the same information. The system can continue operation regularly with all of the remaining devices (usually, just one extra device). With mirroring in place, read requests are balanced among the devices in the array. If any of the devices fail, the device driver marks it as non-operational and stops using it; the driver will continue working with the remaining disks as if nothing had happened. Figure 2 shows how this mode can be used; shadowed regions represent the redundant information.

Next we have the more complex block-interleaved parity (RAID-4) and block-interleaved distributed-parity (RAID-5) personalities. A striping unit is a set of

contiguous sectors. Both the RAID-4 and RAID-5 personalities use one stripe on one of the disks in the array to store the parity information and use the remaining stripes for data.

Whenever one of the data sectors is modified, the parity sector has to be recomputed and written back to the disk. To recompute the parity sector, the device driver has to read the contents of the old parity block and recompute the new parity information. Then it writes the new contents of the data and the parity block.

The difference between RAID-4 and RAID-5 is that the former stores the parity information in a fixed device (the last one of the composing devices in our implementation), while the latter distributes the parity blocks between the composing devices.

Figure 3 shows the RAID-4 arrangement; shadowed regions are the redundant information, which in this case are the parity blocks. RAID-4 has a bottleneck due to the parity disk, since all MD activity depends on the parity bit disk to finish its operation.

Figure 4 shows the RAID-5 arrangement; shadowed regions are the redundant information (the parity blocks once again), and each disk here represents a striping unit. In RAID-5, the RAID-4 bottleneck is eliminated by distributing the work among several disks.

In the case of a disk failure while using the RAID-4 or RAID-5 personalities, the disk array can continue operation since it has enough information to reconstruct any lost disk. The MD device is slower at this point; every access to a data sector in the lost disk requires reading the information from all the sectors in the same stripe as well as the parity bit and computing the contents of the original sector based on this information.

Kernel Changes to Support the New Personalities

The new personalities, as we have implemented them require a change to the ll_rw_blk routine and some extra code in the input/output end-request notification code in the kernel.

The ll_rw_blk routine is modified to make a call to the **md_make_request** instead of calling the kernel's **make_request** when the blocks are scheduled to be sent to an MD device. The md_make_request routine in turn calls the

request-making routine once for each personality to carry out its job. In the case of RAID-0 and the linear modes, md_make_request calls the regular make_request routine.

The ll_rw_blk routine in the presence of the new MD driver is shown here:

```
ll_rw_blk (blocks)
{
    sanity-checks ();
    for-each block in blocks {
        if (block is md-device)
            md_map (block)
    }
    for-each block in blocks {
        if (request-is-for (raid1,raid4,raid5))
            md_make_request (block);
        else
            make_request (block);
    }
}
```

Because of the increased complexity and error recovery capabilities of the new RAID code, the personality code must be informed of the results of its input/output operations. If a disk fails it must mark the disk as non-operational, and in the case of the RAID-4 and RAID-5 code, it must move between the different phases of the process. We modified the kernel input/output end-request routines to call the RAID personality's **end_request** routine to deal with the results.

A typical end_request routine is as follows:

```
raidx_end_request (buffer_head, uptodate)
{
    if (!uptodate)
        md_error (buffer_head)
    if (buffer_head->cmd is READ){
        if (uptodate){
            raidn_end_buffer_io (buffer_head,
                uptodate)
            return;
        /* read error */
        raid_reschedule_retry (buffer_head);
    }
    /* write case */
    if (finished-with buffer_head)
        raidn_end_buffer_io (buffer_head,
                uptodate)
}
```

The Generic MD Changes

We wanted to preserve the behavior of the ll_rw_blk routine so it's as close as possible to what the client code of this routine expects. Since the code now provides error recovery, an innocent and simple-looking request can actually turn out to be a complex set of requests; therefore, the MD code now begins a configurable number of kernel threads used to arbitrate the complex requests. It exports those threads to the new personalities through the **md_register_thread** and the **md_unregister_thread** functions. The MD threads

each sleep on its own wait queue and awake when needed. They then call the personality's thread and run the disk task queue.

The RAID-1 Code

The mirroring personality is the simplest one in the new code. Whenever a read request is made, it is sent to one of the operational disks in the disk array; in the case of a write request, the personality's request-making code puts a write request for each device in the array into the system-request queue.

In the event of a disk error in one of the devices while writing, the device is marked as non-operational, and a message is logged to the syslog facility to notify the operator about the situation. If this error happens during a disk read, then the code retries the read from one of the operational devices; then it puts the read request into a queue and wakes up the **raid1d** kernel thread.

The raid1d kernel thread has just one purpose in its life—to retry read requests. When it wakes up, it retries any queued requests to all of the operational disks.

The RAID-4 and RAID-5 Code

Both RAID-4 and RAID-5 provide block-interleaved parity; the former stores the parity in a single disk from the array, while the latter distributes the parity among all the disks in the array. Most of the code is the same in both modes. Just one routine makes the difference between the two modes.

The easiest code path is the one where all of the disks of the array are working properly. In this case, there are two code paths for the read and write modes. When asked to read a block from the disk array, the code puts the computed location of the data sector in the system request queue and no further complications arise.

In the case of a write, things are more complex since the driver has to write the corresponding block as well as update the parity block. When a single sector from the disk array is written, the code needs to do the following:

1. Read the old contents of the data sector and the old contents of the parity sector.
2. Compute a new parity sector.
3. Write the new data sector with the new parity sector.

Since the upper layers expect the code to put the request on the queue, the code starts up the read requests on the disk and returns to the caller.

When any of the requests have been completed, the **raid5_end_request** is called. This routine together with the RAID-5 thread **raid5d** are responsible for keeping track of the current state of the block. If there are no problems with the disk I/O operations, the request is finally marked as finished and the upper layers can use the block.

If there is an error during block reading or writing, the RAID driver marks the faulting disk as a non-operational disk and continues operation without using it. The driver does reconstruction of lost blocks and computes the parity according to the information available on the other disks. Both block-read and block-write routines become more complex in the cases where something has gone wrong. The disk array is slower, but regular operation of the system can continue until the faulty disk is replaced.

If spare disks are configured on the disk array, the RAID driver starts a thread that re-creates the information on the disk that failed. When it finishes with the reconstruction, the disk is marked as operational, and the driver resumes operation at the regular speed.

Other Changes

We have updated the “userland” utilities that control the MD driver to make use of the features found on the new personalities.

The tool **mkraid** is used to configure a RAID device. It reads a configuration file and creates a RAID superblock, where all the administrative information about the device is stored.

At system boot time, the **syncraid** program checks the RAID superblock to make sure that the RAID system was unmounted cleanly and reconstructs the redundant information in case something went wrong.

There is a notable bottleneck in RAID-4. All of the parity information is kept on a single disk, so any write operation done on any of the data disks in the disk array will incur a write to the parity disk. For this reason, the speed limit in level 4 of the disk array is limited by the speed of the parity disk. It may seem unreasonable to have such a personality, since RAID-5 addresses this problem. We have implemented RAID-4 because, in some configurations of disk arrays, access to the disks may be serialized by the disk controller (this is the case with some IDE drives).

Credits

Future Work

We want to provide a RAID-1+ personality that basically would be RAID-1 plus check pointing; this will let us synchronize the disks at boot time after a disk crash. Unfortunately, PC computers are not shipped with an NVRAM device that would allow us to do this with a minimum amount of work.

The RAID-4 and RAID-5 code can be enhanced so it can determine when a complete stripe is being written to disk. When this is detected, the driver does not need to read any of the old information found on the disk. We can write all of the data sectors as they are found on the request queue, compute a new parity bit sector and add it to the queue as well. This is the preferred method for improving RAID-4 and RAID-5 disk write speed as discussed in "Striping in a RAID Level 5 Disk Array" (see Resources).

Currently, the low-level, request-queue and ordering code writes using the elevator algorithm. Preliminary work in this area shows that the algorithm can be improved in a per-device fashion by algorithms that are aware of the disk geometry. Virtualizing the ordering algorithm for the request queue in a per-device fashion may provide a good performance gain.

When there is a system crash, the RAID-4 and RAID-5 systems need to recompute the parity information on the disk array, since the parity blocks may contain old information. To improve this situation, we want to make use of non-volatile RAM (on those systems that have this kind of hardware) for check pointing the state of the disk array. With this feature, we can recompute the information for any blocks that may have incorrect information when starting up the disk array.

Resources

Gadi Oxman (gadio@netvision.net.il) s the author of the Linux IDE/ATAPI tape and floppy drivers and of ext2ed, a file-system editor for the Linux ext2 file system.

Ingo Molnar (mingo@vger.rutgers.edu) is the author of several hacker-type kernel patches; he is a Linux kernel "must read".

Miguel de Icaza (miguel@gnu.ai.mit.edu) is one of the GNU Midnight Commander authors. He also worked on the Linux/SPARC kernel port.

Archive Index Issue Table of Contents

Advanced search

Disk Hog: Tracking System Disk Usage

Ivan Griffin

Issue #44, December 1997

Using a Perl script and World Wide Web, Mr. Griffin shows us how to determine which users are hogging the disk space.

A job that most system administrators have to perform at one stage or another is the implementation of a disk-quota policy. Being a maintainer of quite a few machines (mostly Linux and Solaris, but also AIX) without system enforced quotas, I need an automatic way of tracking disk quotas. To this end, I created a Perl script to regularly check users' disk usage and compile a list of the biggest hogs of disk space. Hopefully, in this way, I can politely convince people to reduce the size of their home directories when they get too large.

The **du** command summarizes disk usage for a given directory hierarchy. When run in each user's home directory, it reports how much disk space that directory is occupying. At first, I wrote a shell script to run du in a number of user directories, with an **awk** back end to provide nice formatting of the output. This method proved difficult to maintain when new users were added to the system. Users' home directories were unfortunately located in different places on each operating system.

Perl provided a convenient method of rewriting the shell/awk scripts into a single executable, which not only provided more power and flexibility but also ran faster. Perl's integration of standard Unix system calls and C-library functions (such as **getpwnam()** and **getgrname()**) makes it perfectly suited to this sort of task. In this article, I will describe how I used Perl as a solution for my particular need. The complete source code for my Perl script is available by anonymous download in the file ftp://ftp.linuxjournal.com/pub/lj/listings/issue44/2416.tgz.

The first thing I did was make a list of the locations in which users' home directories resided and put this list into a Perl array. For each subdirectory of the directories in this array, a disk-usage summary was required. This summary

was obtained by using the Perl **system** command to spawn off a process running du.

The du output was redirected to a temporary file using the common $$ syntax, which is replaced at run time by the PID of the executing process. This guaranteed that multiple invocations of my disk-usage script (while unlikely) would not clobber each other's temporary working data.

All of the subdirectories were named after the user who owned the account. This assumption made life a bit easier in writing the Perl script, because I could skip users such as root, bin, etc.

I now had, in my temporary file, a listing of disk usage and a user name, one pair per line. I wanted to split these up into an associated hash of users and disk usage, with users as the index key. I also wanted to keep a running total of the entire disk usage and the number of users. Once Perl had parsed all this information from the temporary file, I could delete it.

I decided the Perl script would dump its output as an HTML formatted page. This allowed me great flexibility in presentation and permitted the information to be available over the local Intranet—quite useful when dealing with multiple heterogeneous environments.

Next, I had to decide which information I needed to present. Obviously, the date when the script ran is important, and a sorted table listing disk usage from largest to smallest is essential. Printing the *GCOS* (general comprehensive operating system) information field from the password file allowed me to view both real names and user names. I also decided to provide a hypertext link to the user's home page, if one existed. To do this, I extracted their official home directory from the password file and added the standard, user directory extensions to it (typically, public_html or WWW).

Sorting in Perl usually involves the use of the “spaceship” operator (**<=>**). The **sort** function sorts a list and returns the sorted list value. It comes in many forms, but the form used in my code is:

```
sort sub_name list
```

where **sub_name** is a Perl subroutine. **sub_name** is called during element comparisons, and it must return an integer less than, equal to or greater than zero, depending on the desired order of the list elements. **sub_name** can also be replaced with an in-line block of Perl code.

Typically, sorting numerically in ascending order takes the form:

```
@NewList = sort { $a <=> $b } @List;
```

whereas sorting numerically in descending order takes the form:

```
@NewList = sort { $b <=> $a } @List;
```

I decided to make the page a bit flashier by adding a few of those omnipresent colored ball GIFs. Green indicates that the user is within allowed limits. Orange indicates that the user is in a danger buffer zone—no man's land—in which they are dangerously close to the red zone. The red ball indicates a user is over quota, and, depending on the severity, multiple red balls may be awarded to truly greedy users.

Finally, I searched, using all the web-search engines, until I found a suitable GIF image of a piglet, which I placed at the top of the page.

Figure 1. Disk Hog Screen Shot

The only job left was to arrange to run the script nightly as a **cron** job. This job must be run as root in order to accurately assess the disk usage of each user—

otherwise directory permissions could give false results. To edit root's cron entries (called a **crontab**), first be sure you have the environment variable **VISUAL** (or **EDITOR**) set to your favorite editor, then type:

```
crontab -e
```

Add the following single line to any existing crontab entries:

```
0 0 * * * /home/sysadm/ivan/public_html/diskHog.pl
```

The format of crontab entries is straightforward. The first five fields are integers, specifying the minute (0-59), hour (0-23), day of the month (1-31), month of the year (1-12) and day of the week(0-6, 0=Sunday). The use of an asterisk as a wild card to match all values is permitted, as is specifying a list of elements separated by commas or a range specified by start and end (separated by a dash). The sixth field is the actual program to be scheduled.

A script of this size (with multiple invocations of du) takes some time to process. As a result, it is best scheduled with cron—I have it set to run once a day on most machines (generally during the night, when user activity is low). I believe this script shows the potential of using Perl, cron and the WWW to report system statistics. I have also coded a variant of it that performs an analysis of web-server log files. This script has served me well for many months, and I am confident it will serve other system administrators too.

This article was first published in Issue 18 of LinuxGazette.com, an on-line e-zine formerly published by Linux Journal.

Ivan Griffin (ivan.griffin@ul.ie) is a research postgraduate student in the ECE department at the University of Limerick, Ireland. His interests include C++/Java, WWW, ATM, the UL Computer Society (http://www.csn.ul.ie/) and of course Linux (http://www.trc.ul.ie/~griffini/linux.html).

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #44, December 1997

Our experts answer your technical questions.

Setting Up Linux to use Two Processors

I just upgraded to a dual Pentium Pro machine. The BIOS sees the two processors but I can't tell if Linux (Kernel 2.0.27) uses both processors. How do I make Linux use its SMP features?

—Jon Bishop Red Hat 4.1

To take advantage of SMP you'll need to recompile your kernel to use both processors. First make sure the kernel-source RPM is installed. Then go to /usr/src/linux and edit the Makefile. You'll find a line that looks like:

```
# SMP = 1
```

Uncomment that line by removing the hash symbol. Then build a kernel as you normally would (see the Kernel HOWTO at http://sunsite.unc.edu/LDP/ for more information).

—Donnie Barnes, MIS Director Red Hat Software, info@redhat.com

Signal 11 Error

When I try to compile a new kernel with custom specifications, the compile fails at approximately the mid-point, giving me an error message that reads:

```
gcc: internal compiler error 11.
```

Please tell me what this could mean.

—Joe Ortiz Slackware 2.0.30

It very likely has something to do with your hardware configuration. There is a web server dedicated to the signal 11 problem. Check out http://www.bitwizard.nl/sig11/. Hopefully the information there will solve your problem.

—Pierre Ficheux Lectre Systèmès

I have seen this happen on several machines. Every time the problem is a bad memory chip installed on the computer's motherboard. Signal 11 is the Segment Violation error which generally occurs when a program's pointers go awry.

gcc is a stable enough program that this should not occur during a simple compile, but it does use memory heavily, and Linux (or any Unix) is quite sensitive to bad memory.

It is also possible that a library file on your system is corrupt or incompatible with other libraries. If, however, this was the case, you should see problems in other programs as well. Library corruption should only happen if you have recently installed new libraries or modified your existing system libraries.

—Chad Robinson, Senior Systems Analyst BRT Technologies, chadr@brt.com

Mixing Linux and NT

How can I get Linux to install on a system along with WinNT? I have plenty of unpartitioned space available.

—Casey Woodrum Red Hat 4.2

The only trick with mixing Linux and NT is configuring the boot process. Linux will install as usual, but you may need to fix up the boot setup by hand once the installation is finished. Assume NT is on partition /dev/sda1 and you install Linux on partition /dev/sda2. Try to be sure your Linux partition is a primary partition on a cylinder less than 1024. This isn't necessary, but it simplifies things.

You should use Microsoft's master boot record instead of LILO. Do not install LILO onto the master boot record of the hard disk. NT will crash dump in most cases with LILO as the master boot record. With all this in mind, take the following steps:

1. Edit the /etc/lilo.conf file so that:

```
boot=/dev/sda2
```

is the first line. Selecting /dev/sda2 as the target for LILO during installation should have done this for you.

2. Add an entry to boot NT at the end of lilo.conf:

```
other = /dev/sda1
label = nt
```

If the installation program is smart enough, you should be able to configure this during installation.

3. Run LILO to install the new lilo boot configuration.

4. Reboot into NT. Use FDISK under NT to mark partition 2 (/dev/sda2) as active. You should get LILO on reboot. From the LILO boot prompt, you can type **nt** to get the NT boot loader.

If the installation process doesn't configure LILO correctly, you may need to boot Linux from floppy and edit the files on the hard disk from floppy.

—Larry M. Augustin, VA Research lma@varesearch.com

An undocumented FDISK parameter can help you out of tight spots if you create trouble on your disk. Run:

```
fdisk /mbr
```

to restore the master boot record on your boot drive. This will remove LILO, allowing you to use Windows NT if you somehow cause trouble during the installation process.

—Chad Robinson, Senior Systems Analyst BRT Technologies, chadr@brt.com

Configuring Red Hat for a Second PPP Connection

I cannot get a second PPP connection to work correctly. I used the X Window tool **netcfg** to set up ppp0 to connect with the University of South Florida. This setting seems to work properly, connecting and disconnecting fine. Then, I set up ppp1 to connect to Compuserve. This appears to connect okay, but I cannot shut down the modem without shutting down my Linux PC. When I look at the system messages file, I see that the scripts start out using ppp1 settings for the connection but once PPP is started it says it is connected to ppp0. I also see that ppp0 registers with the kernel and not ppp1 as it should. I have tried to figure out the various scripts involved but can't make heads or tails of them. I am new to Linux and don't understand the BASH scripting language well. Help.

—Mike Richards Red Hat 4.2

Red Hat's PPP number assignment scheme is broken. Their scripts don't guarantee that ppp1, labeled in the setup utilities, will be activated in the kernel as ppp1. This happens because PPP connections are assigned dynamically. Although you may have defined ppp1 in the Red Hat configuration utility, that setup will be registered as ppp0 inside the kernel if it is the only PPP connection active.

I need a little more detail about your setup to suggest the best fix to your problem, but here's how the scripts work under Red Hat. This may give you enough information to work out a solution.

The file /etc/rc.d/init.d/network is run at boot time with the argument “start”. It looks for configuration files of the form ifcfg* in the /etc/sysconfig/network-scripts/ directory. As a result, if you edit (as is common when not running X) one of those files with an editor that creates a backup file (e.g., ifcfg-eth0~), the Red Hat scripts will run the backup file as well as the new file. For each connection type (Ethernet, PPP, PLIP, etc.), there are “up” and “down” files that start and stop the connection. The network script runs the “up” script with the appropriate ifcfg file as an argument, e.g., /etc/sysconfig/network-scripts/ifup-ppp /etc/sysconfig/network-scripts/ifcfg-ppp0.

There is a corresponding “down” script to shutdown the connection called /etc/sysconfig/network-scripts/ifdown-ppp /etc/sysconfig/network-scripts/ifcfg-ppp0.

You can run these scripts from the command line for testing purposes.

—Larry M. Augustin, VA Researchl mai@varesearch.com

My Mouse Won't Work

My new system came with a new Microsoft Mouse. This mouse doesn't work with Linux. After a couple of days of messing around, I've come to the conclusion that there is a problem with my model of the Microsoft Mouse—on the underside of the mouse is the designation Serial Mouse 2.1A. I have concluded that the problem lies with this mouse because the new system works perfectly well with my old Microsoft Mouse. Is something wrong with the new mouse?

—Ed Green Slackware 2.0.29 Walnut Creek

I don't have access to the newest mouse, so Francois Chastrette has helped a lot with this problem. We are working on a satisfactory solution to include in

gpm 1.12 right now (end of August). By the time you read this *Linux Journal*, the new version of the mouse server should be available by FTP.

X support might take a bit longer as the X team has a huge package to manage. In the meantime, you can use the **-R** option of **gpm** to feed clean mouse packets to the X server.

—Alessandro Rubini rubini@linux.it

Archive Index Issue Table of Contents

Advanced search