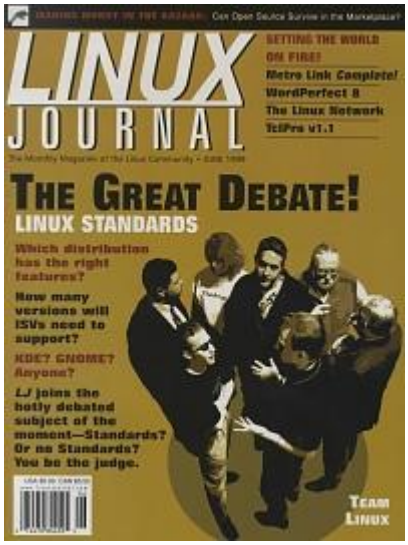
Linux Journal Issue #62/June 1999



Mr. Tharan tells us how to use the PPR spooler for large networks.

Linux in Schools *by Rob Bellville*

How a K-12 school system is using Linux to supply a myriad of stable network services to its students and staff.

Linux for Enterprise-Resource Planning *by Uche Ogbuji*

Mr. Ogbuji takes a look at enterprise resource planning and Linux's place in this market.

Linux Web Server Toolkit *by Keith P. de Solla*

A review of the LINUX Web Server Toolkit, a book that takes the reader completely through the procedure of building a web server.

Archive Index

Advanced search

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

Focus: Standards

Marjorie Richardson

Issue #62, June 1999

This month, we look at the Linux standardization efforts.

Standards—a fairly innocuous word that seems to create a storm of controversy whenever it is spoken. Everyone agrees it is a good thing, but no one agrees on what standards should encompass or how they should be enforced. Whether for auto parts or operating systems, standardization can be a big plus for the consumer.

The Linux operating system, unlike other software products, has multiple sources—each distribution represents a different implementation. The differences are generally in the installation software and methods (RPM vs. DEB packages, for example); however, nothing is currently in place to prevent a company from adding a feature to the operating system and still call it Linux. Other companies are free to adopt the feature, but this is not required.

This month, we look at the Linux standardization efforts. Two things are very clear in the standards debate:

- Distributions want to remain unique in order to maintain marketplace advantage.
- Users and manufacturers of applications software (ISVs) want applications that will run on whichever distribution they own, i.e., they want applications to run on all distributions.

These two things are not mutually exclusive. After all, users do not want one distribution to become the Linux “Microsoft” (it might be one other than their favorite), so users too are all for uniqueness in distributions. And no distribution wants to be the odd man out—the distribution on which a major application doesn't work; so, the distributions also are for compatibility. Developers more than anyone want standards that will enable them to write

programs that will work across all distributions without hassle. Thus, it appears as if all sides have a common ground on which to meet.

Setting and following standards is the only way to ever ensure cross-distribution compatibility for applications. However, standards that are defined in a rigid and finely detailed manner will be ignored by developers as unrealistic and difficult to follow. Finding that optimum position between standards that are too lax and those that are too rigid is the laudable goal of the Linux Standards Base Project. Dan Quinlan, the project leader, tells us about the plans of the LSB in his article in this issue.

To find out where all the distributions stand on this issue, Norman Jacobowitz talked to representatives of each by e-mail and at the LinuxWorld Expo. Some were more forthcoming than others; see who said what in Norman's article this month.

Want to express your opinions? Join the discussion groups on *Linux Journal Interactive*, http://interactive.linuxjournal.com/.

—Marjorie Richardson, Editor in Chief

Archive Index Issue Table of Contents

Advanced search

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

The Past and Future of Linux Standards

Daniel Quinlan

Issue #62, June 1999

“The nice thing about standards is that there are so many of them to choose from.” --Professor Andrew S. Tanebaum (author of MINIX).

Despite their well-earned reputation as a source of confusion, standards are one of the enabling factors behind the success of Linux. If it were not for the adoption of the right standards by Linus Torvalds and other developers, Linux would likely be a small footnote in the history of operating systems.

The Early Days

Some people believe the interest in Linux standards is very recent, precipitated by the upswing in commercial interest. However, before Linux was even named, conformance to an open standard was an important goal. Here is one of the first postings by Linus Torvalds about a project that would soon be named Linux:

```
From: torvalds@klaava.Helsinki.FI
        (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: Gcc-1.40 and a POSIX-question
Date: 3 Jul 91 10:00:50 GMT
Due to a project I'm working on (in minix),
I'm interested in the POSIX standard definition.
Could somebody please point me to a (preferably)
machine-readable format of the latest POSIX rules?
FTP sites would be nice.
```

A month later, Linus posted:

```
As to POSIX, I'd be delighted to have it,
but POSIX wants money for their papers, so that's
not currently an option.
```

Despite the high cost of a copy of the POSIX standard in 1991, it became one of the primary standards for Linux. POSIX, the Portable Operating System Interface, is a standard application programming interface (API) used by Linux and many other operating systems (typically UNIX and UNIX-like systems).

There are several major benefits to using the interface defined by POSIX. It makes it easier to write source code that can be *compiled* on different POSIX systems. It also gives Linux application developers and Linux kernel developers a well-defined API to share. That means application developers don't need to track most kernel changes as long as the kernel continues to behave as POSIX says it should.

In addition, using POSIX as the API for Linux enabled Linus and other early Linux developers to use existing free programs written by the GNU Project, the BSD operating system and many other free programs which adhere to the POSIX specification.

It is important to note that POSIX does not provide for precompiled binary applications to be run on any POSIX operating system. Since it provides source code compatibility, but not binary compatibility, POSIX is often thought of as a *source standard*. Early development of Linux was made under the Minix operating system. In fact, Linus originally wanted to make Linux binary-compatible with Minix. That idea was dropped when the differences between Linux and Minix became too great, but some traces of the Minix heritage of Linux still linger here and there.

When asked about the importance of POSIX in early Linux development, Linus Torvalds said, “Linux started out *very* aware of POSIX, but even more so of the unofficial de facto standards.” He elaborated that, at the time, these de facto standards were approximately SunOS (the precursor to Solaris) behavior, somewhere between BSD and System V. “Basically, I wanted to not have to spend too much time porting user mode programs (it wasn't something I was all that interested in), and POSIX helped in that,” Torvalds said.

Linux Today

POSIX.1 (the POSIX kernel interface) is still considered by Linus Torvalds and other kernel developers to be the “base standard” for the kernel. Some later additions to the POSIX specification have not been as useful as POSIX.1, and design issues often have to be resolved before Linux can make use of a standard. Linus describes one of the best examples of such a design decision:

> Often there are standards that are too generic to be very useful as a guide for the kernel. The “pthreads” POSIX threads standard is one example: some people have tried to implement their kernel threading model according to it, and the standard simply is not very well suited to that.

> Even in that case, I still wanted to support the standard; I just did not want to *natively* implement the

> standard as-is. So Linux **clone** was created: the infrastructure to do threading under Linux, on top of which you can implement pthreads and also other threading models.

It is difficult to list all of the other standards used today by Linux. TCP/IP, Ethernet and other formal and de facto standards form the basis for networking in Linux. The IBM PC is one of the best examples of a de facto standard. (This standard is now more formalized.) The PC allowed thousands, and then millions, of people to run Linux on a system typically used for Microsoft Windows. It is a lot easier to take over the world if you run on the standard hardware of the day. Just as significant, the GNU C compiler (without which there would be no Linux) is built on top of the K&R (Kernigan & Ritchie) and ANSI C standards.

Common Implementation

In addition to formal standards like POSIX and ANSI C, one of the most prevalent types of standards used by Linux systems is a *common implementation*. The best example, of course, is the Linux kernel. No written specification is available for many of the interfaces provided by the kernel. However, no specification is needed because only one strain of kernel is accepted by everyone (not just all distributions or all developers, but truly *everyone*). Want to know the standard? Read the kernel source.

While not always quite as clear-cut, many more examples of common implementation standards are accepted by the entire Linux community, or at least a large chunk of it. Take the utilities and programs provided by the GNU Project. For example, it has long been accepted that the utility **find** is the enhanced GNU version, not the BSD one or some other version. If you are writing an application for Linux only, it is probably safe to rely on most of the functionality provided by GNU find. Because so many GNU utilities like this are used, Richard Stallman, founder of the GNU Project, insists that people refer to Linux as “GNU/Linux” or as a “Linux-based GNU system”. This tends to upset many people, but most developers seem to recognize Stallman's immense contribution and don't make a big fuss about it, even if most of them still say “Linux”.

The GNU project's share in the success of Linux does not end there. Linux has now standardized on the GNU C library (glibc) as the common implementation for future Linux systems. The glibc project aims for compliance with several standards. According to Ulrich Drepper, the GNU libc maintainer, all of the important standards are supported as long as they don't conflict with common sense. That list includes ISO C 90, ISO C 9x, POSIX and UNIX 98.

Filesystem Hierarchy Standard

What if there is not a common implementation, a formal standard, or even an informal, de facto or ad hoc standard? One thing you can do is try to pick the best and most common practices and base a standard on them.

The Filesystem Hierarchy Standard (FHS) was among the first written standards developed specifically for Linux. The FHS aims to standardize the locations where files and directories are placed in the system. Standard locations are needed so that applications can be compiled and run well on different Linux distributions; it is also helpful for developers. If you want to write documentation for Linux, or if you need to work on more than one variety of Linux, it is invaluable knowing you can expect to find important files and directories in a standard location.

If you are wondering why a standard such as POSIX could not be adopted, the reason is one did not exist. Each UNIX vendor had its own solution, and there was a great deal of overlap, but the specifications and rationale for each vendor's layout were either lacking or nonexistent. Unfortunately, no multi-vendor standard was available for a standard layout of files and directories.

The FHS was started in 1993. At the time, the Linux OS was approximately two years old. As he does today, Linus dictated how things would work in the Linux kernel and did not exert much influence on other areas of the Linux operating system. With no central authority for those other areas, a number of Linux distributions had considerably different layouts. Some were like the BSD operating system, others were more like SunOS, and others were different still. (Incidentally, most of the names of the big distributions back then were different from the ones we hear today: Debian, Linux/PRO, MCC, Slackware, SLS, TAMU, Yggdrasil, etc.)

The first version of the FHS was based on ideas from SunOS 4, SVR4, 4.3BSD, 4.4BSD, SunOS 4, HP-UX and many other UNIX systems, but it was also based as much as possible on common practices distilled from the various Linux distributions existing at the time. The completed specification attempted to take the best of each file-system layout and combine them into a solution that worked well.

Since then, subsequent releases of the FHS have been refinements of the original specification, importing a few additional ideas from other areas (including 4.4BSD and SVR4).

Today, most Linux distributions follow FSSTND 1.2 (the precursor to FHS 2.0). They do this not because they are forced to, but because it cleanly addresses a

problem they were having. FHS 2.1 should be available by the time this article is published.

Linux Standard Base

The widespread availability of third-party commercial software for Linux is a relatively new phenomenon. Many vendors are faced with a problem they have never seen before.

At one time, all the software on your Linux box generally came from one of two places. Either it came with the distribution, or someone (you, a friend, maybe even a system administrator) compiled it from the source and installed it locally. Distributions are very good at integrating software and making sure everything works together. Assuming someone had the expertise, it is possible to get even a sub-standard program compiled and installed for your Linux boxes.

What if someone else wants to give you a pre-compiled binary package to install on your system? They are forced to consider which distribution you run, what version of the distribution, your kernel version, and a slew of other considerations before they can be reasonably certain that the application will install correctly on your system and work. If the goal is to provide a binary package for everyone, it is even worse. You must tailor the application for any and all distributions, and then compile, build and test it on each one, too. Is it any surprise that few vendors support even three or four of the major Linux distributions for their applications?

One of the more common fears about Linux is that *fragmentation* will occur because it uses an open-development model. As in real life, fragmentation occurs when something breaks into many different pieces which are no longer connected. Fragmentation is not a new phenomenon in the free-software community. Sometimes it happens for good reasons, perhaps when a maintainer is not doing a good job. Sometimes it happens for less worthy reasons, such as a personality conflict between lead developers.

This has happened before. A graphical-oriented version of the GNU Emacs editor called *XEmacs* split off of the version maintained by the Free Software Foundation (FSF) because of technical differences between the FSF and the developers of Lucid Emacs (the predecessor to XEmacs). The XEmacs FAQ does not take a rosy view of the situation:

> There are currently irreconcilable differences in the views about technical, programming, design and organizational matters between RMS [Richard Stallman] and the XEmacs development team which

> provide little hope for a merge to take place in the short-term future.

> If you have a comment to add regarding the merge, it is a good idea to avoid posting to the newsgroups, because of the very heated flame wars that often result.

When there is a split like this, it usually has the most impact on the end user. Add-on packages work with one variety of the software, but might not work with another. That is more or less what happened to GNU Emacs, although developers try to compensate for it as best they can.

A small amount of fragmentation, such as the difference between Linux distributions, is good because it allows them to cater to different segments of the community. Because Linux is Open Source, different distributions have the freedom to be unique. For example, the Extreme Linux distribution targets high-performance clusters of Linux PCs running the Beowulf clustering software. Rather than a single all-in-one distribution, each distribution can target a segment of the Linux community and try to meet that segment's needs better than any other distribution. Users benefit by getting a Linux distribution that more closely meets their needs than is possible under a single distribution model.

Also important is the attitude about fragmentation in the Linux community. Everyone is concerned that fragmentation could become a problem, and wants to ensure that applications can run on any variety of Linux. That is where the Linux Standard Base (LSB) is involved. The LSB project is working to define a common subset of Linux that everyone can count on, independent of the distribution. By defining only what can be expected in a minimal base Linux system, the LSB is attempting to find a balance between stifling Linux development and the possibility of Linux fragmenting into several totally incompatible versions.

The main problem the LSB is addressing is that software vendors must port and test their software on multiple Linux distributions, because even a small difference between distributions can result in major problems for the software when the vendor's software has been based on the behavior of one distribution. The difficulties ultimately affect everyone: users, developers, application vendors, Linux companies, et al.

Aside from the danger of fragmentation (which, by the way, Linux has avoided for the last eight years without an LSB), there are secondary dangers, namely FUD—fear, uncertainty, and doubt. The LSB hopes that by making fragmentation a remote possibility, it will help bolster confidence and win

support for Linux from even the most conservative sectors. How will it do that? The Linux Standard Base Goals are as follows:

- Enable software applications to run on any LSB-compliant distribution.
- Increase compatibility among Linux distributions.
- Help support software vendors and developers to port and write software for Linux.

Those are the things the LSB developers want to do. On the other hand, we want to avoid some things. Our LSB Guidelines for Success include:

- Don't tread on distributions—everyone wants them to be unique.
- Do no more than what is required to solve the basic problem of application portability.
- Don't break old systems or prevent future advances.

LSB intends to generate a written specification for what distributions should provide and what application vendors can expect in a base system, and provide a test suite and sample implementation of the base system.

Linux and Future Standards

Is it enough for Linux developers to make their own way based on standards developed by outside groups such as the IEEE, The Open Group, ISO and ANSI? Probably not. Linux developers have been able to pick and choose which standards to adopt and how to implement them, but as standards are revised and extended, Linux developers want to ensure future standards also meet their needs.

One such revision in progress is a joint revision of the POSIX standards by the IEEE, The Open Group and ISO. The group revising the standard is known as the Austin Group. Unlike previous POSIX standards, the goal is a common set of documents shared by all three organizations. USENIX, the Advanced Computing Systems Association, is helping to fund two Linux developers to attend meetings and participate in the revision. The two developers are Ulrich Drepper, the glibc maintainer, and H. Peter Anvin, author of the kernel automounter and maintainer of the Linux device list. The POSIX revision, Ulrich says, will throw away or at least make optional some of the less wanted parts of the old standards (such as STREAMS). This is a good thing for Linux because those parts have not been adopted by the entire Linux community. The result is that fuller compliance with POSIX will become more likely.

In addition, Ulrich adds, there are functions he would like to see standardized in the new POSIX specification. Some of those function specifications may come

directly from the glibc project. If that happens, maybe some future operating system can put some of the standardization blame on Linux.

Resources

Daniel Quinlan (quinlan@transmeta.com) is the chair (i.e., project leader) of the LSB, the editor of the Filesystem Hierarchy Standard and a member of the Linux International technical board. He is employed as a System Administrator at Transmeta Corporation. Outside work, he is currently getting into indoor rock climbing.

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

The Distributions Take a Stand on Standards

Norman M. Jacobowitz

Issue #62, June 1999

Mr. Jacobowitz talks about standards with representatives of the various distributions by e-mail and at the LinuxWorld Expo.

With all of the recent debate about standardization and the future of Linux standards, it's natural to wonder where the major Linux distributions and other key industry players stand on the subject. So, to find out, *Linux Journal* posed the following question: “What stance does your organization take on the subject of Linux standards and the growing standardization issue?” We received some very interesting responses with quite a variation in length from the major vendors. Here they are, in no particular order.

Red Hat Software, Donnie Barnes

We at Red Hat are generally in favor of good standards that speak to existing as well as future problems. History has shown that adherence to standards can be very important and failing to do so can cause fragmentation that leads to a very unhealthy situation for all players involved. We do our best to make sure we adhere to useful standards such as the FHS (Filesystem Hierarchy Standard).

New standards for Linux are evolving, and we are working hard to be a part of that evolution. Standards are very important in maintaining compatibility across all Linux distributions. Compatibility is important to keep independent software vendors (ISVs) from seeing Linux as a fragmented mess that can't be supported. ISVs are important because they bring the applications to Linux. Applications are important because they bring users to Linux. As you can see, it's a trickle-down effect that has large implications.

That said, we are working hard to make sure the evolving standards are good ones. Bad standards can also be very harmful. So far, the Linux community has been very good at working out good standards. With a responsible community effort from all players, I fully believe the Linux community can maintain that

trend. If so, you can expect Red Hat to participate fully with the standards that are created.

Caldera Systems, Ransom Love

Thank you for the opportunity to respond on Caldera Systems' feelings about Linux Standard Base (LSB). Everyone probably has a slightly different definition of what they think LSB is and what it should be. I will take the liberty of telling you what I think it should be. LSB's main objective should be to solve the problem facing commercial ISVs. LSB should be an effort that would enable commercial software developers to port once and have their applications run on all major Linux distributions. If it does not solve this problem, Linux will have a major problem in the not too distant future. Nearly every ISV we have spoken to wants to remain “distribution agnostic”, but they do not want to support four or more versions to accommodate Linux. The current answer seems to be one of three options: publish the source code to the application and allow the community to support the different versions, pick a kernel version and a library version and allow the Linux providers to offer patches, or just pick one of the distributions to support.

I will address the problems with all three of these ideas. First, publish the source code to the application. This will work in some cases but not in the majority. Some of these industry players have millions if not billions of dollars tied up in proprietary software and hardware that will not allow them to publish their solutions. They do and will continue to support Linux and the open-source effort, but their board of directors, attorneys and stockholders will not allow them to publish source code at this time and may never allow it. Publishing the source opens up significant liabilities. Netscape published the browser source only to outsource it to another company. Publishing source has to be done after significant thought and consideration.

As to the second option of allowing the ISV to pick a version of the kernel and library, what if Oracle picks one and Informix picks another? Or if Corel and Oracle pick different versions and the customer cannot run one of these solutions without searching the Internet to find patches? How will Linux compete against Microsoft when they've claimed to be a one-stop shop? The commercial customer loses because he must know his way around to find the patches and fixes on the Internet. Even VARs and Systems Integrators who are technical enough to do so cannot afford to spend the time.

The third option is to port to just one distribution. This seems counterintuitive to the Linux and Open Source community that believes in choice. However, there are those who would support this option to solve the problem. The major concern with porting to only one distribution is you lose out on the innovations being made by others who are targeting different markets. Some of the leading

distributions have targeted the developer as the principal customer, as developers have made up the majority of those purchasing Linux until now. Features like time-to-market are of vital interest to the developer. Packages are gathered from many different sources around the Internet to ensure the quantity and timeliness of delivery, and little concern is given to ensure that source and binaries match. After all, the developer knows enough to make it work.

Commercial concerns cannot deal with more than two releases a year, so time-to-market is not a feature but a liability. Caldera Systems has been focusing on the commercial or business market from its inception and has built features like self-hosting into the distribution. Self-hosting is the discipline to match all source to the binaries. It allows the commercial entity to set all of the compilation parameters for the entire distribution, maximizing performance and support options while limiting security and liability concerns. All the packages are tested and integrated as a single whole. Distributions focused on the developer may choose to take the time to add more packages which appeal only to the developer, or release a technology before it is stable so that developers have easier access to it. The bottom line is that the Linux community and the business customer will lose options and choice if a single distribution is used as the reference platform. Some ISVs may choose one and some another. The only real answer is for the Linux community and the current providers of Linux to collaborate.

Linux and the whole Open Source community have a great opportunity to become significant players in the commercial environment if they remain open-minded. If Linux and the Open Source community can make it easy for commercial players to interact with Open Source without incurring legal liabilities or major costs, Linux will be a significant player and the customer will win.

LSB can be a major step toward solving this problem and showing the industry that Linux is different, not only because it is published under an open-source license, but because its providers have vision. If the major Linux distribution providers can collaborate and support a common kernel and library version as well as the basic interfaces that enable an ISV to port once to Linux and support all Linux providers, everyone wins. If the Linux providers are irresponsible and choose not to cooperate but continue on with their own agendas, Linux is not as likely to fragment, but will fall significantly short of its potential to be the predominant computing platform, open source or otherwise, in the industry.

Linux is not as likely to fragment, as its cousin UNIX did in the OSF/UNIX International debacle, because it is Open Source. However, being Open Source will not completely solve the problem either. The premise that the best open-

source technology always wins is not true. GNOME and KDE are prime examples. KDE is excellent technology that has matured faster than GNOME. The main reason GNOME was started was because the Qt libraries KDE depends on were not Open Source. Since Troll Tech has now published the libraries under an accepted open-source license, all concerns about KDE should be over. Instead, a significant public relations effort has begun to continue to spread uncertainty and doubt on KDE. Technologically, it is superior in all cases and is more mature. Why would someone persist in promoting false information about KDE? Because they have invested time, money and energy in GNOME and the KDE developers do not have the funds to fight back. Again, the Open Source community is not immune to the influences of the almighty dollar.

Clearly, if Linux is to achieve its potential as a major industry alternative, we cannot rely upon Open Source alone. All of the industry players must act responsibly. Caldera Systems has always wanted Linux to be a viable commercial alternative, so we believe the only real answer is to support LSB.

Pacific HiTech, TurboLinux, Scott Stone

Pacific HiTech will take an active role in the development of Linux standards and will make compliance with the adopted standards a design goal of all our Linux products. We believe a well-defined set of Linux standards is important so that ISVs can more easily port their applications to the Linux operating system without worrying about distribution compatibility.

I'd like to see them develop to the point where any compliant distribution will have the same shared libraries available and the same basic file-system structure. Some people have suggested that a standard package manager would also be important, but I'm not necessarily sure that's true—for example, .deb and .rpm can both provide a system in which a given third-party application could work, provided that the shared libraries and path structure are the same. What I *would* like to see is a common system for “registering” applications so that desktop managers and other programs would have a standard method of determining what is installed on the system.

Slackware, Patrick Volkerding

Slackware is waiting to see what the proposed standards are before we commit to complete compliance, but I do think the effort is a step in the right direction for Linux. I'd really like to see a list of standard version numbers to use when building shared libraries—that should be a simple first step in getting binary compatibility among the distributions back on track. However, the standards should probably not go into such detail that all distributions end up looking cut from the same cookie cutter. It is the different design philosophies which make

different Linux distributions appeal to different kinds of users. It'll be interesting to see how these issues are balanced.

Debian, Nils Lohner

The general stance of Debian is that standardization is a good thing. Dale Scheetz is a Debian developer and actively involved in the LSB work. LSB is also an SPI-supported project. As Linux continues to grow, it will become more and more important for vendors producing software to know it will run on a number of different distributions without requiring distribution-specific versions. The growth of Linux will be greatly helped by applications being ported to run on it, and this growth must be supported as much as possible.

Debian, Wichert Akkerman

Debian has been involved in the LSB project from the beginning. One of our developers, Dale Scheetz, is working on the LSB right now. We've been talking with Red Hat since before LSB so that we can develop binary compatibility between the distributions. The important system libraries should be fully compatible across all distributions. We don't want to see the kind of incompatibilities suffered by users of 16-bit MS Windows software when they upgraded to 32-bit MS Windows. Debian is currently working to adopt the File Hierarchy System, but we feel a few issues remain to be resolved. Debian is happy to help create standards and compatibility with the other distributions; the LSB is one of the methods we are using to make this happen.

SuSE, Marc Torres

SuSE has been a member of the Linux community since 1992, and we have people dedicated to and working on the LSB project. We are proud to be active members of this project. We are hoping to see some sort of minimum library standards because that is what our customers want. ISVs need that kind of standard to work efficiently. Hardware vendors deserve a standard in order to certify their hardware across distributions.

Stampede Linux, David Haraburda

It's my personal opinion that standardization is good thing. But like always, too much of a good thing can be a bad thing. It really all depends on which aspect of the Linux system you want to standardize, and how much. Where do you begin and where do you stop? If we standardize the file system layout, packaging and user tools, is anything left? I'm not sure of LSB's plan—but from my reading, it doesn't look like they have a well-laid-out plan quite yet. Standardizing file locations is a great idea; I know it would solve half of the problems I encounter. I read somewhere LSB plans on having a standard

packaging system. That is a tricky subject for me to even personally comment on—we have a lot of plans for our Stampede Linux Packaging system (SLP). There are some features the standard may not include that we'd like to have. Obviously, we can contribute to the project ourselves, but if the majority doesn't like our idea, then it won't go in. The great thing about Linux is that if you don't like something, you can't complain. Don't use it, find something else, hack the source code, or start a whole new project. If a situation like this arises, the user who wants a feature can no longer search for another package; he must try and somehow convince the authors to add it.

At this time, two extremes are apparent: one is “everyone does their own thing, their own way” and the other is “everyone does what the standard says”. I'd like to think we aren't at either of these extremes right now. That fine line between them needs to be found, and when someone finds it, I'll be able to comment further.

Currently, with the information we have, Stampede's stance is neutral, but this could easily change. It could change tomorrow, next week, next month or next year (just like any Linux-related project). For now, I will try to keep a close eye on what goes on and keep others of the Stampede development team informed. It will be much easier to make a decision once a clearly defined set of rules has been set.

Conclusion

Well, there you have it—the positions of the major distributions on the topic of Linux standards. As you can see, all positions have one major theme in common: a serious concern for the future of Linux Standards and a desire to see standards work for every user.

Norman M. Jacobowitz is a freelance writer and marketing consultant based in Seattle, Washington. He can be reached at normj@aa.net.

Archive Index Issue Table of Contents

Advanced search

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

WordPerfect 8 for Linux

Michael Scott Shappe

Issue #62, June 1999

If you're already perfectly happy with WordPerfect 7, in that you're not experiencing any problems and not missing any particular feature, you may simply not need this upgrade.

- Manufacturer: Corel Corporation(ported by Software Development Corporation)
- E-mail: custserv2@corel.com(info@sdcorp.com)
- URL: http://www.corel.com/(http://www.sdcorp.com/)
- Price: Server Edition: $495 USPersonal Edition: $69.95 US (MSRP),$49.95 via Linux Mall, Linux Central and Linux Systems Labs
- Reviewer: Michael Scott Shappe

WordPerfect 8 for Linux is an evolutionary improvement over last year's WordPerfect 7 port to Linux, and so, appropriately, this review is merely a follow-up to my review of WP7 (“Word Perfect 7 for Linux”, April 1998).

To summarize that review, I'd found WordPerfect 7 to be a very good port of a solid product. It had all the features I needed, plus many features I didn't, which are necessary for buzzword compliance. It fit handily into what was then a very low-end machine: a Pentium 120 laptop with 16MB of memory and 1GB of disk space, only 600MB of which were allotted to Linux.

My problems with it at the time were few. It was a bit on the slow side, particularly under Xfree86; I found I had to turn off many of the dynamic features, like Spell-as-you-go and the application bar at the bottom of the window. Under AcceleratedX, performance was a bit better, but I still had to turn off the dynamic spell-checking to get good speed. I also found the style system overly complicated, and some crashes in the help system and the HTML conversion code.

Small Steps

How does this next version stack up? Is it worth the download time or the $50US for the shrink-wrapped version? I'd like to be able to say, “Yes, definitely!” Instead, I find myself thinking, “It depends.”

If you're already perfectly happy with WordPerfect 7, in that you're not experiencing any problems and not missing any particular feature, you may simply not *need* this upgrade. I know that's a bit heretical in the modern software industry, but it's true. Most of the features already in place have had only minor improvements.

Some nice new features are present for novices, including an extensive guide system called “PerfectExpert”. Grammar-as-you-go is new and useful if you like grammar checkers (I actually do).

The speed is certainly improved. When I was first given access to early betas, I still had the same low-end configuration and found that not only could I leave Spell-as-you-go on, but I could also leave the new Grammar-as-you-go active. The newer version keeps up just fine with my typing. The final release seems to be equally snappy, which made me very happy.

Getting What You Pay For

Several features are available only in the purchased version, not in the download. Notably, the ExpressDocs template feature is not available in the download. The number of fonts available is extremely limited. The “Insert->Shape” menu works, but, “Insert->Graphics->Draw A Picture” does not; so you can insert shapes “in-line”, but not a complete picture unless you draw it with a different program. Also, the ability to create and insert charts and graphs is disabled.

Lingering Problems

One of my biggest complaints from WordPerfect 7 remains a complaint with its successor; indeed, it appears that no work whatsoever has been done to improve it. The style-sheet system is still incredibly complex—almost too complex to be useful. It is difficult to define styles that are part of a library of styles rather than part of a specific document, and defining individual styles is a cryptic process.

The interaction of printers and fonts is also a bit strange. In most word processors, the list of fonts available does not reflect the printer selection; instead, fonts are translated or substituted at the print-driver level. In WordPerfect, the list of available fonts is a direct reflection of the way the

currently selected printer is set up, and setting up a printer is a less-than-obvious process. This is as much a flaw in UNIX, which has no nice, neat, centralized print model, as it is in WordPerfect.

To Buy or Not to Buy

If you need or want a good commercial word processor for Linux, WordPerfect 8 is definitely a good value. It works, and it works well. It's also inexpensive, which is always good.

If you already have WordPerfect 7 and are satisfied with it, there's probably no great rush to upgrade. On the other hand, the upgrade won't break the bank, either. Actually purchasing the product will give you a wider range of fonts and access to all the features, as well as save you considerable download time.

As much as I was hoping for more out of this upgrade, I'm still very happy with the product and recommend it. I hope Corel continues to have success with it.

Michael Scott Shappe is a somewhat frazzled software engineer for AetherWorks Corporation, a start-up in Saint Paul, Minnesota. When not writing reviews or copy editing for *Linux Journal*, he's reading—and attempting to write—fiction, or attending Society for Creative Anachronism events. He can be reached at Mikey@Hundred-Acre-Wood.com, and his web page is at http://www.14850.com/web/mikey/.

Archive Index Issue Table of Contents

Advanced search

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

Metro Link Motif Complete!

Liam Greenwood

Issue #62, June 1999

Metro Link *Complete!* is an excellent package for anyone developing in Motif on Linux.

- Manufacturer: Metro Link
- E-mail: sales@metrolink.com
- URL: http://www.metrolink.com/
- Price: $149 US
- Reviewer: Liam Greenwood

Metro Link Motif *Complete!* is a runtime and development version of Motif 1.2, 2.0 and 2.1 for Linux x86 and Linux on Alpha. It includes a bonus suite of Motif widgets for x86.

To install the software, you need either an x86 Linux machine or an Alpha Linux machine with 8MB to 16MB or more of main memory and between 14MB (for runtime-only) and 80MB (full installation with all options) of disk space. Both libc5 and glibc versions for x86 are available, and the manual is very specific, not only on what levels of other software you need, but also which commands you use to find out.

I received one CD-ROM and a draft copy of the manual. The product certainly didn't suffer from being a pre-production version, with no glaring errors in the rather nice manual. The *User's Guide* primarily covers the installation and configuring of the runtime environment (MWM, etc). If you want to do development, you will need to either find documentation elsewhere or be content with what's on the CD.

The installation was done on an Alpha station running Debian 2.0 and a Pentium running Caldera OpenLinux 1.2. The installation on the Alpha went smoothly, with only the text stopping in mid-sentence in a dialogue box causing

any disturbance. The x86 installation went reasonably well also. Metro provides an easy-to-use GUI installation program, as well as instructions for doing it on your own. They provide both RPMs and tar archives; I chose the RPMs. The GUI leads you through the questions on which versions you want to install, selects the appropriate RPMs and installs them. Metro runs the RPMs in a text widget which allows them to display the output text from the RPM. In my case, I had two package conflicts, xbmbrowser-5.1-1 and Pixmap-2.6-1. These caused an RPM to fail to install; however, the GUI doesn't scan the displayed text for errors, so when you move to the next screen, nothing tells you the installation has failed. I think this is a flaw in an otherwise great installation, and Metro Link should have the GUI clearly display whether the installation was successful on completion.

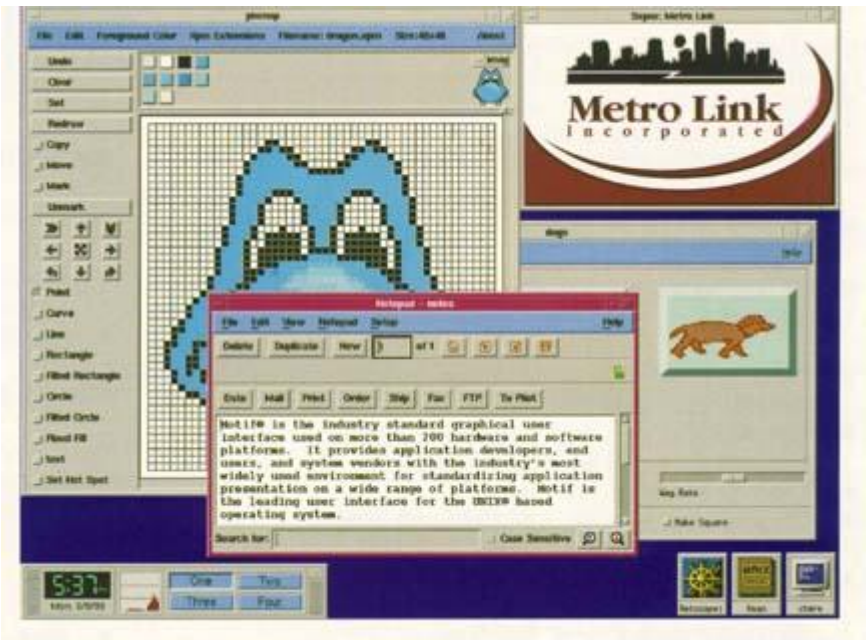


Figure 1. Screenshot

On the x86 system, all of the runtime elements worked, including MWM (Motif Window Manager) fpanel (like the CDE, common desktop environment, panel) and the Motif version of an xterm. (See Figure 1.) They didn't run very well, however; X took around 45% of the CPU and fpanel 52% while maintaining a load average of just under two on an otherwise idle machine. Also, fpanel and MWM both require editing text files to configure them. With KDE and GNOME making big efforts to provide GUI administration tools for the desktop, this feature is unlikely to find favour. On the other hand, this package is unlikely to be attractive to someone who just wants a CDE-like environment. On the Alpha, MWM and fpanel ran fine, but the Motif xterm wouldn't run, as Debian didn't have a libncurses. After linking /lib/libncurses.so.4.2 to /lib/libtermcap.so.2, it ran fine.

The demo programs, stored in the /usr/src/motif directory, built and ran fine on both platforms, as did xmcd.

The software comes with a bonus package from the KL Group Inc., for x86 and Motif 2.1 only, a complete and fully functional but unsupported version of the XRT Professional Developer's Suite family of Motif widgets. This requires an additional 50MB of disk space.

The package comes with a whole swag of PostScript documentation on disk, with most of the OSF documentation included for each level of Motif. For 1.2, they include:

- programGuide: OSF/Motif Programmer's Guide
- programRef: OSF/Motif Programmer's Reference (man pages)
- releaseNotes: OSF/Motif Release Notes (excerpts)
- styleGuide: OSF/Motif Style Guide
- usersGuide: OSF/Motif User's Guide

Version 2.0 and 2.1 also have the widgetGuide: OSF/Motif Widget Writer's Guide. Also provided on the CD-ROM are the User's Guide and the Alpha and x86 READMEs (formatted release notes), all in PostScript format.

Metro Link *Complete!* is an excellent package for anyone developing in Motif on Linux. The quality of the package is high, and I'm confident the few rough edges are due to the pre-release version I received for review.

My thanks to Martin Lucina for running this on his AlphaStation.

Liam Greenwood (liam@sasquach.gen.nz) is a Solutions Architect at EDS (NZ) Ltd. He has been using Linux since Slackware was making 0.99 kernels into distributions. He is both pleased and dismayed that the viable ports are growing faster than he can wangle platforms to run them on.

Archive Index Issue Table of Contents

Advanced search

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

TclPro v1.1

Daniel Lazenby

Issue #62, June 1999

This tool suite includes four related tools: debugger, checker, wrapper and compiler.

- Manufacturer: Scriptics Corporation
- E-mail: info@scriptics.com
- URL: http://www.scriptics.com/
- Price: $1000 US per named user
- Reviewer: Daniel Lazenby

TclPro is a collection of tools that should make your Tcl/Tk programming life a little easier. This tool suite includes four related tools: debugger, checker, wrapper and compiler. Some tools are graphical, and others are command line. Two TclPro-specific interpreters, **prowish** and **protclsh**, are also included. These four tools provide several valuable services.

The first tool is a Tcl compiler. The current release of Tcl compiles source code before execution. With TclPro, one can compile source code independent of execution. This gives you the ability to distribute your programs in a compiled format. Not all of the Tcl source can be compiled with TclPro v1.1; items which cannot be compiled independent of execution will be compiled when executed. A procedure that takes a script as an argument is an example of something that cannot be compiled before execution.

Distributing a Tcl/Tk program to non-Tcl platforms requires the distribution of several Tcl/Tk files and libraries besides the application's Tcl script. With the TclPro Wrapper, you can bundle all of the various files into one statically or dynamically wrapped file. Wrapping a Tcl application statically creates a stand-alone bundle of **tclsh** and all related Tcl libraries and application files. I wrapped a simple 1820-byte file and in return received a file almost 1.67MB in size.

Programs prepared in this manner can be loaded and run on platforms without regard to the installed Tcl version.

Wrapping an application dynamically will reduce the application's size. The same dynamically wrapped 1820-byte file produced an output file of about 107.5KB. There is a cost for this smaller footprint. Dynamically wrapped applications require the target platform's Tcl installation to be compatible with the Tcl release of your application. Both non-compiled and compiled Tcl files may be fed to the TclPro Wrapper.

Each new version of software presents some new functions or features. Sometimes a little backward compatibility is lost among those new features and functions. I am notorious for dropping a semicolon or curly brace in my code. TclPro Checker addresses these and several other programming issues. It can detect parsing and syntax errors. Four types of warnings are provided by TclPro Checker. There are warnings about platform portability of the code, performance-optimization opportunities for code segments, potentially incorrect command usage, and warnings about changes in syntax conventions between older and newer Tcl versions. The output of TclPro Checker streams across the screen. You may want to page the output or redirect it to a file.

Figure 1. TclPro Debugger Screen

The fourth tool is a graphical debugger with a trick up its sleeve. Figure 1 shows how information is presented in the three panels. The upper-left panel presents the stack. Variables and their values are presented in the upper-right panel. Code being debugged is displayed in the lower panel. A toolbar provides the means of stepping in, through, around and over procedures and related code.

TclPro Debugger's other useful feature is remote debugging. With this tool, you can actually debug a Tcl/Tk program residing on another platform. Don't get too excited just yet. You cannot randomly select any remote file to debug. Some prep work is required before files can be debugged remotely. Three commands with arguments must be entered into the remote file. Only after doing this can you debug them remotely.

Install Experience

The message here is not that I had an installation problem. It is about the timeliness and accuracy of Scriptics e-mail support staff. They knew me only as another person who was having a problem installing their fully-functioning download product for evaluation on a Caldera OpenLinux (COL) v1.3 Linux platform.

Installation materials and the Scriptics web site said TclPro was known to install and function properly on Solaris, HP and Irix UNIX flavors. SuSE 5.3+ and Red Hat 5.0+ were the only two Intel Linux distributions listed as known to support TclPro. The instructions said TclPro should work on other Linux distributions, providing they used glibc2. Having just upgraded my platform to COL v1.3 with glibc, I could not see any reason why TclPro should not run on my platform.

After verifying libc's installation and reading all of the available README and INSTALL.TXT files, I tried my first TclPro install. It failed. I repeated the install and verified my steps and the displayed error messages. I contacted Scriptics Support with the symptoms and error messages. The next business day, I received a response requesting some additional information, a basic explanation of the install process and a couple of things to try. Later in the day, I sent Scriptics Support the requested information. I received a response the next business day. The folks at Scriptics quickly spotted that COL v1.3 appended a dot (.) to the end of TclPro's CD-ROM file names. This dot was appended only to file names that did not already contain an extension. I did not have a solution, yet I knew what the problem was and knew an answer was being sought. Another business day passed and I received a workaround for my COL v1.3 installation. Using the workaround, the TclPro1.1 installation went flawlessly.

I was set up to use KDE. My screen resolution is such that the entire install dialog boxes did not appear on the screen. Resizing the dialog boxes under KDE prevented access to the dialog box buttons. I suggest using the X Window System for product installation. With X Windows, you can move portions of the dialog box off the screen to see the various buttons.

Documentation and Other Resources

TclPro1.1 comes with a 107-page User's Guide in both hard- and soft-copy formats. The soft copy is in PDF format and requires the Acrobat 3.0+ viewer. Local on-line documentation also includes browser-based help and man pages. Additional information, resources and links are available at http://www.scriptics.com/resources/.

I compared a couple of the User's Guide chapters with the browser's help content. The User's Guide and the local browser-based help content are similar in some respects and also contain some differences. One primary difference is the availability of message IDs within the on-line help.

I did notice a couple of minor things about the browser-based on-line help. The on-line “Using TclPro Checker” topic referred to a nonexistent example. I did find the same topic and example in the User's Guide. On the other hand, I attempted to use an example command from the book and it failed. The

command usage error messages and the on-line help quickly pointed out the book's error.

Licensing

At the time of this writing, Scriptics' web site and other literature indicated TclPro is licensed to a named individual and not the platform. No clear indication was given as to how licenses could be reallocated within an organization if a personnel turnover were to occur. Licensing relief may be on the horizon. The next release of TclPro will offer a UNIX-based network license package. A five-user license will be the smallest available network license.

Support

A 30-day e-mail installation and evaluation support service is included. As mentioned above, my experience with this support was quite positive. Scriptics also offers three levels of fee-based support. The lowest level is an annual product update service. This level of support provides only product updates. The two other levels of support, Gold and Platinum, go beyond product updates and are sold on a per-user basis. At the time of this review, the product update service is a prerequisite for either the Gold or Platinum support levels—not an unreasonable condition. I feel some of the other conditions on support need to be rethought. For example, the Gold-level support required purchasing support for a minimum of five users. This is rather expensive if you own only one or two licenses.

Minimum Platform

I was unable to find recommended minimum system requirements in the documentation or on the web site. This product's command-line tools run on a 486/66 platform with 24MB of RAM. I do not recommend running the debugger on this class machine unless you do not mind waiting several minutes. TclPro1.1 runs quite nicely on a 300MHz Pentium II with 64MB of RAM.

Conclusion

I found TclPro Version 1.1 easy to learn and found value in each of the four TclPro tools. I feel the product is definitely worth the download and evaluation. The User's Guide very adeptly describes TclPro: “TclPro is an evolving piece of software. We will continually improve TclPro according to the user feedback and the needs of the Tcl community.” Let Scriptics know if it does not meet your needs.

Note that a beta release of TclPro Version 1.2 was made available shortly before this article was submitted for publication.

Daniel Lazenby (dlazenby@ix.netcom.com) holds a BS in Decision Sciences. He first encountered UNIX in 1983 and discovered Linux in 1994. Today he provides support for a range of platforms running Linux, AIX and HP-UX.

Archive Index Issue Table of Contents

Advanced search

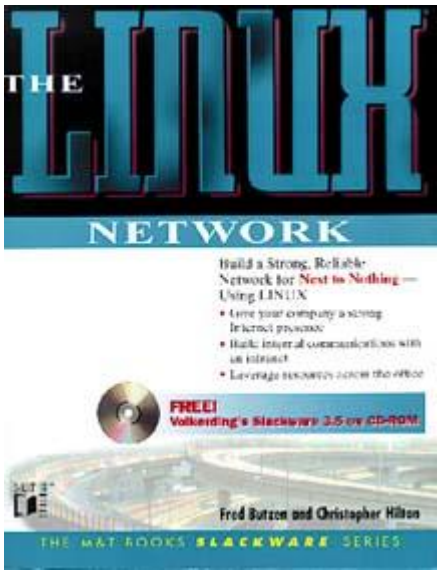
Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

The Linux Network

Duane Hellums

Issue #62, June 1999

The Linux Network is chock full of networking gems that reflect decades of experience and significantly ease the task of creating a Linux-based network.



- Authors: Fred Butzen and Christopher Hilton
- URL: http://www.idg.com/
- Publisher: IDG Books Worldwide, Inc.
- Price: $39.99 US
- ISBN: 1-55828-589-X
- Reviewer: Duane Hellums

Administrators considering networked Linux servers and workstations can now find almost everything they need in one place and no longer have to rifle through mounds of books and technical articles in numerous magazines. *The Linux Network* is chock full of networking gems that reflect decades of experience and significantly ease the task of creating a Linux-based network.

This book is a “primer” for individuals or smaller offices and organizations that typically cannot afford solutions offered by consultants and proprietary off-the-shelf UNIX servers. Readers should be aware that the book is specialized for networking, so it covers only adding or modifying networking options to Linux, *not* basic installation of the Linux operating system and kernel. The book includes Slackware version 3.5 based on Linux kernel release 2.0.29, and as such, does not discuss configuration of other distributions such as Red Hat, Caldera, SuSE or Debian. Of course, other than a few minor exceptions, the underlying technologies are extremely similar and the discussions pertain to most Linux distributions.

The authors provide detailed instructions on how to implement typical server functions such as FTP, TELNET, e-mail, news, NFS and SMB file sharing, web server and browsers, proxy server, “gateway” router and DNS. They also include some cool tools, such as details on how to connect to 3Com's 56Kbps telephone line test system, how to optimize Linux networking for security, how to effectively use the syslog daemon and other network debugging tools, and how to configure the point-to-point protocol (PPP) daemon and dial daemon for “dial-on-demand” connections to the Internet. They include a comprehensive networking primer and share many networking jewels, such as the limitations of the **ping** tool, the differences between IP masquerading and SOCKS, and scalability estimates in terms of users and available network bandwidth. One thing they do much better than many Linux books is include information on how to configure networking to meet the needs of several different real-world examples. They also show *all* of the Slackware networking configuration options with detailed descriptions of each choice, whereas most writers include only those items on the “typical” installation decision tree, often without any useful description of the options.

The Linux Network provides almost all of the tools needed for an administrator to create a networked Linux workstation, workgroup server, or gateway in either an intranet or Internet environment. I say *almost* because it lacks a few small items, such as detailed alternatives to UUCP mail service, how to configure Linux and Windows to provide an RS-232 serial port “terminal” connection to a Linux server, how to configure a Linux server to allow dial-in modem connections for remote users, and details on how to implement a Linux DHCP (dynamic IP allocation) and news server. The latter two items were deemed to be outside the scope of the book. UUCP is quite limiting, and many small businesses may have application needs which require remote dial-in, or call for a simpler “network” of Windows-based text terminals connecting to a Linux server, either through a serial port or a multiport serial card.

The writing style in *The Linux Network* is straightforward and not overly cute, unlike some contemporary technical writing (with the minor exception of the

use of the pronoun *she* exclusively to denote *all* users and administrators in the book, taking a needlessly political stand on a moot issue while simultaneously abandoning centuries of practical English language custom). The authors include pertinent, useful and interesting details and elaboration where appropriate. Information is nicely organized in a logical manner, usually as a series of steps which show how to install, configure, activate, test and debug networking software and tools.

Emphasis on heterogeneous networking solutions using Linux and Windows is adequate, especially where it involves file and printer sharing. However, coverage of Windows e-mail and newsreaders is somewhat out-of-date and limited to Microsoft Exchange client software. Administrators of Linux networks that include some Windows desktops in the mix should give serious consideration to using Microsoft Outlook Express, which receives no coverage in this book. As with most Linux books, unfortunately, this one has the usual smattering of Windows-bashing. There seems to be an overall bias towards the Netscape Internet suite (mail, news readers and web publishing tools) over Microsoft's (Explorer, Outlook Express, FrontPage Express), which does make some sense for companies with both Linux and Windows desktops that want to standardize applications across both platforms. There also seems to be a preference for Linux across the enterprise, to include Linux workstations rather than the more likely Windows desktops.

The book emphasizes using the Linux server to handle both local *and* Internet mail services, primarily through UUCP connections, which the authors admit is generally either prohibitively expensive or not available in the administrator's area. For a typical small office sharing a dial-on-demand Linux Internet gateway among half a dozen or so machines and users, a more affordable and graceful solution might be to use individual Internet e-mail accounts through an Internet Service Provider, and local Intranet e-mail accounts on the Linux server (a solution for which Outlook Express excels).

All in all, *The Linux Network* is a very useful book for any administrator who has Linux up and running on a PC, but is considering the benefits of networking Linux with other PCs and possibly the Internet. For those people who do not already have Linux installed, this book should be combined with a good general purpose Linux book containing detailed basic installation and configuration instructions. Many people run Microsoft Windows (Workgroups, 95 or 98) on desktop PCs and want to integrate them with a Linux workstation or two, or use a Linux server as an affordable, efficient, “open” alternative to an expensive, proprietary NT server. Those people should supplement this book with a good basic Windows TCP/IP networking book. Actually, Microsoft's Internet site active server pages and application wizards make it fairly simple to download, install, configure and use Microsoft's suite of Internet tools.

Duane Hellums is a program manager, software engineer and IT consultant with a Master of Science in Information Systems degree and nine years of network and system administrator experience. His e-mail address is duane@hellums.com.

Archive Index Issue Table of Contents

Advanced search

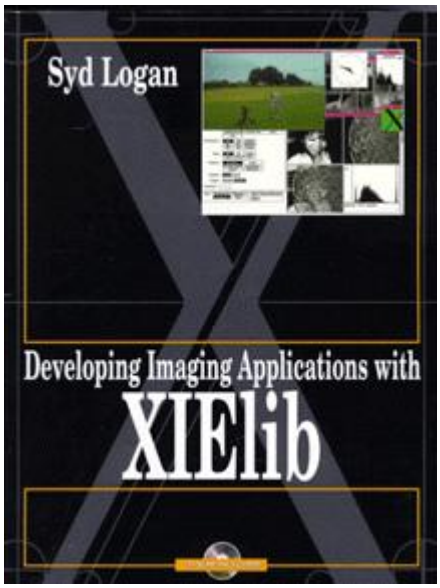
Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Developing Imaging Applications with XIElib

Michael J. Hammel

Issue #62, June 1999

The X Image Extension, more commonly known as XIE, provides a mechanism for graphic image management in the client/server environment of X11.



- Author: Syd Logan
- Publisher: Prentice Hall
- URL: http://www.prenhall.com/
- Price: $54 US
- ISBN: 0-13-442914-1
- Reviewer: Michael J. Hammel

Computer graphics have come a long way for Linux users in the past two years. Applications of all kinds have started to show up in many places, coming from both the Open Source community as well as commercial vendors. One reason for this recent growth is the flexibility and extensibility of the native windowing system: the X Window System, also known as X11. Although end users are the eventual benefactors of this flexibility, it is the development side of Linux that

will be most interested in one of X11's least-known extensions—the X Image Extension.

The X Image Extension, as the name implies, is not part of the core X11 distribution, although any system using X11 Release 6 (aka X11R6) should have this extension included. The X Image Extension, more commonly known as XIE, provides a mechanism for graphic image management in the client/server environment of X11. Image decompression, for example, can be handled on the server side (the display side) instead of overloading clients with this common and resource-consuming task.

Some time back, the author of *Developing Imaging Applications with XIElib*, Syd Logan, contacted me about this extension and recommended the GIMP developers take a look at it. At the time, I was modestly involved in GIMP development, back in the very early days when the GIMP was still a Motif application. XIE never really seemed to catch on in the developer community for the GIMP. Perhaps it was simply because no text was available from which to learn the API (the applications programming interface or the interface into XIE's software library). It may also have been due to the fact that not all developers were on Linux at the time and not all UNIX vendors had migrated to X11R6, a situation that continues to this day.

Despite the GIMP not using XIE, I've still been interested in what XIE is all about. What can it be used for? How difficult is adding it to an existing application? Does it help solve some of the more mundane problems in computer graphics, or is it more focused on specific types of problems?

Syd's book is well-organized and provides answers to most of these questions. The text is thick—well over 600 pages—with chapters ranging from introductions to existing image handling methods in Xlib to the nuts and bolts of XIE's API and its underlying data structures. The chapters flow from an overview of the XIE architecture into high-level concepts such as “photoflo elements” and image file formats to more detailed descriptions of using “process elements” to do format conversions and the use of “techniques”, the algorithms used by the server side for image processing.

Even with my self-taught experience in graphics terminology, I discovered XIE has a whole new set of terms to learn. A fairly complete glossary of these terms is included at the end of the text, but you'll find Syd's definitions within the text to be a better source of information. He does a good job explaining new terms quickly and fitting them into the larger picture that is XIE.

Accompanying the text is a CD with a modest bit of sample code from the text. The README on the CD is quite extensive and should be read before trying to

build or run any of the software on the CD. Some of the files on the CD do not have read permissions for anyone but the file owner, so you'll have to copy the files from the CD as the root user and then do a **chown**/**chmod** sequence to get them installed properly.

The libtiff and jpeg libraries, required to build the XIE samples, don't build easily from the CD. I downloaded the latest libtiff source from the FTP sites specified in the README file (found in the libtiff directory on the CD) and replaced the old version with the new version. I couldn't do that with the JPEG library, because the main archive location of ftp//ftp.uu.net/ doesn't allow connections from dial-up hosts whose names don't resolve in DNS. So, I grabbed the source from the GIMP server (ftp//ftp.gimp.org/pub/libs/). The new versions built much more easily. I did not, however, install these libraries on my system, as I already had working versions installed. The /xiesamples directory hard codes the library paths to these image libraries (libtiff and libjpeg) to be relative to the samples directory. I simply built the libraries and then renamed the default directories (from the archive files) to the directories expected by the XIE CD software. After this, the CD software built fine.

The sample software under the /xiesamples directory covers most, if not all, examples from the text. Since all of the sample clients on the CD were written on Linux, they all built fine (after updating the two libraries described previously). Programs here are small examples of the base features of XIE as described in the text. From the README file on the CD:

> Most of the samples work with gray-scale JPEGs, a few work with CCITT images stored in TIFF files, PGM, PPM, color JPEGs, or raw SingleBand (gray-scale) images.

Unfortunately, I had problems running these sample programs on my system. After rereading the README file, I discovered some of the samples work only on 8-bit PseudoColor visuals. I have my system running in TrueColor mode. Take a look at the README file for explanations of each sample; it will tell you if you can specify a visual and/or the type of color map to use. However, in my limited tests, the only sample program I was able to run under a TrueColor visual was the **backendtest** program.

Chapter 2 has a fairly good discussion on using Xlib for image manipulation for anyone who had problems understanding it from the O'Reilly *Xlib* manuals. This text is a programming guide, not a reference guide. To my knowledge, a printed reference guide for XIE is not yet available.

Throughout the text, Syd provides examples of how to make XIE API calls, including well-commented data structures used in the API. It's obvious Syd understands the ins and outs of the API. He provides extensive source code

examples—I hardly went through three pages without seeing at least one snippet of code.

Plenty of gray-scale example images are printed in the text, and at least a few of these lose some of their impact without color. An eight-page color gallery is in the center of the book. However, these images are useful only for explaining some general graphics terminology and techniques and are not really descriptive of XIE itself.

I found Chapter 15 to be one of the best chapters simply because of its application to other graphics areas. Color space conversion is a common task and one that is not well-supported by many Linux graphics applications. For example, three paragraphs into this chapter, I learned that

> YCbCr images represent color and luminosity as separate channels in the image. Humans are more sensitive to changes in luminosity than to changes in color, [so] compression algorithms can apply greater compression ratios to the color bands than are applied to the luminance band (Y) without degrading the perceptible quality of the image.

This sort of information, while explaining why certain color spaces are supported by XIE, also provides definitions that are applicable to other color management tools.

One issue not covered well in this text is how XIE might be used, if it can be used, with tools like OpenGL or Mesa. It might not make sense to combine the two, but there doesn't seem to be any mention of how XIE fits into the overall X11 collection of core features, extensions and third-party graphics libraries.

Summary

Syd Logan's text is an extensive work that thoroughly covers the X Image Extension. Since Syd worked on a sample implementation of XIE, he is well-qualified to discuss the topic. The question is whether XIE is the right solution for your imaging tasks. Its ability to offload some work to the server can be very beneficial. Granted, my graphics experience is not as extensive as it could be, but with the runtime size of some X servers already growing to over 20MB on x86 Linux, one has to wonder if XIE is a good idea or not. A thorough reading of *Developing Imaging Applications with XIElib* should help you answer that question.

Michael J. Hammel (mjhammel@graphics-muse.org) is a Computer Science graduate of Texas Tech University, and a software developer specializing in X/Motif. Michael writes the monthly “Graphics Muse” column in Linux Gazette, maintains the Graphics Muse web site and the Linux Graphics Mini-HOWTO, helps administer the Internet Ray Tracing Competition and recently completed work on his new book, *The Artist's Guide to the GIMP*, published by SSC, Inc. His outside interests include running, basketball, Thai food, gardening and dogs.

Archive Index Issue Table of Contents

Advanced search

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

MiniVend—the Electronic Shopping Cart

Kaare Rasmussen

Issue #62, June 1999

If you need a catalog system for your web page, this product may be just what you are looking for.

When managing an ISP, you will eventually need a solution for electronic commerce. Many possible solutions are available, but the problem is that they are either big and expensive or small freeware that does not include all the necessary options. I heard about a system called MiniVend, released under the GNU General Public, and decided to try it out.

Features

MiniVend is a full-featured electronic catalog system (commonly known as a shopping cart) with on-line ordering capability. It is designed to provide an interface complete with SSL security and full database support.

Some of the main features of MiniVend 3.0 are:

- Multiple catalogs allow one server to run many shops, so it is ideal for an ISP.
- Security is provided through SSL for credit card ordering and PGP for mailing of orders.
- It has a well-developed database integration with SQL support, including ODBC.
- A very powerful search capability is provided with fast binary search, range searching, numeric and alphanumeric search sorting with reverse, numeric and case-insensitive options, etc.
- All aspects of the appearance can be controlled. MiniVend supports frames, and the pages can be built on the fly or pre-built for heavily used items.

- It is very flexible, with sales tax, discount and freight calculation, easy price adjustments and much more.
- Cookie support allows users to leave the shop and come back without losing session state. It works well with all browsers and includes CyberCash support.
- Easy administration is possible with automated installation and configuration, and off-line and on-line database builds.

MiniVend is a client/server system. The browser talks with a small application that in turn talks with the MiniVend server through a socket. For this reason, you don't have to load the MiniVend server for each user session, which might overload the system.

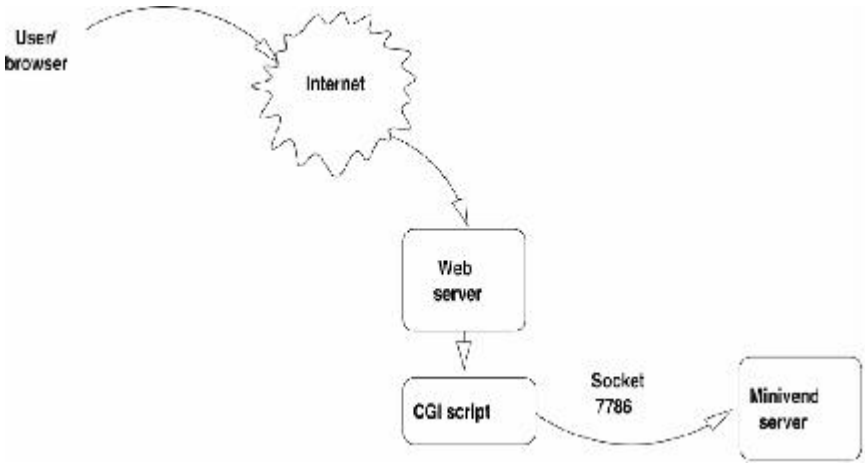


Figure 1. The MiniVend System

I don't know how the name came to be—there is nothing “mini” about it. It is a full-featured electronic commerce system that can meet the needs most people have for such a system. MiniVend is powerful and correspondingly complex. It can easily scale from a few items per catalog to a million items or more, with excellent performance. If you have only a few items and don't intend to grow, MiniVend is probably overkill.

Requirements for Electronic Commerce

The key issues are ease of use and flexibility. The system should do what is needed without too much administrative work and also include all your desired features in an easy-to-use manner.

Options that are a must include:

- The ability to exchange data between internal and external databases. The product information will normally be kept in a company database of some kind, and entering data by hand would be cumbersome, to say the least.

- Automate the order process as much as possible. The ideal situation will be one where you only have to feed the order into your system, with all taxes, freight costs, etc. handled by the software.
- A method of handling discounts for selected customers and for volume sales should be available.
- Feedback should be sent to the customer by e-mail when an order is made. This helps to catch any errors made by the system or the customer.
- Security for payments is also an issue. The system needs to use SSL and perhaps another encryption protocol when it sends data to and from the database.
- Good documentation and support is needed. The documentation should be so well-written that it can get you started quickly (in a matter of hours); the support should get you past any show stopper.

We all have different needs, and each person will have a preference as to what goes into a system for electronic commerce. I like a system that does only what it is supposed to do: handle product information, searching and ordering, while leaving domain name registration and chat rooms to their own specialized tools.

Why I Chose MiniVend

Whenever I have a choice between commercial and Open Source products, I always try out the freeware first. Of course, there is the issue of price, which in the case of Shopping Carts can be a very big issue. Access to the source code will in most cases guarantee that the system is more error-free, because users can have a look at the code and suggest solutions to any existing problem. After all, the users are the people who know where the trouble is, and they are normally more motivated to find and correct it.

However, it is not only a choice between commercial and Open Source systems. Other Open Source products are available, but MiniVend stands out in the features category. It is simply incredible how much functionality is provided and how easy it is to configure. All the configuration options I could think of and more are included.

Also, MiniVend is easy to use with several merchants, and I can use it with my Apache 1.3 with no problems.

Installation

Installation couldn't be easier. Simply type

```
su wwwrun
tar xvzf minivend-latest.tar.gz
```

in the directory where you want the installation directory to be created. This command will unpack the latest version of MiniVend. The version I downloaded was 3.11, which was released only a week or so before I tried it. (See Resources.)

The user wwwrun owns the web server's /DocumentRoot directory. Any user with write access to both this and the /cgi-bin directory can be used.

```
./configure
```

The installation process includes a long question-and-answer session, but you can just press enter to answer most of the questions; MiniVend comes with sensible default values. When the installation of MiniVend is finished, you will get the opportunity to install a catalog:

```
-> /home/httpd/mvend
-> Make the simple demo now? [yes]
```

Select a short, mnemonic name for the catalog. This will be used to set the defaults for naming the catalog, executable and directory, so you will have to type in this name frequently. If you are doing the demo for the first time, you might use “simple”:

```
Catalog name? simple
```

The rest of the installation consists of some simple questions I won't repeat here. The server name and /cgi-bin location should be working before you try to install MiniVend.

Figure 2. The Simple Demo

When everything is installed and ready, you can start the MiniVend server with this command (use the path to your installation):

```
/home/httpd/mvend/bin/start
```

Installation Notes

MiniVend is a big program, meaning it is very complex, and if you encounter problems, support is not always available. A mailing list is available, but none of my questions got answered during the test period. That included questions about internationalization and the installation problem I had on the one machine on which it was needed.

On the other hand, I found the documentation to be very extensive, though a bit hard to follow at times. The 600KB HTML documentation (see Figure 3) is a definite plus.

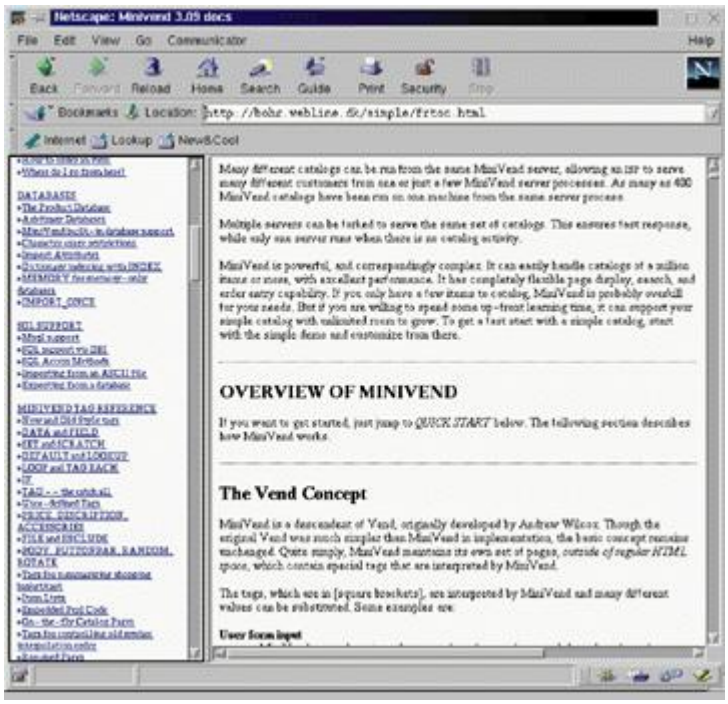


Figure 3. MiniVend HTML Documentation

Be very careful with your choice of user for the MiniVend server. This user must have write access to both the /DocumentRoot and the /cgi-bin directory, as you will install a demo with both HTML pages and CGI-scripts. But don't make the mistake of running the MiniVend server and the web server as the same user -- that is a security risk to avoid. The best solution in a production system is to create a new user to handle the MiniVend server.

It is the user's duty to ensure the security of sensitive data. You can set up MiniVend to leave the data anywhere on the server or send it by e-mail to the

merchant, but you must consider how to encrypt the data. MiniVend supports PGP encryption, so there is a way to set up a secure data transfer.

During this short test period, I obviously didn't have time to test for “wear and tear”, but everything seems to be designed with ease of use in mind. The system tables, e.g., sales tax or freight values, are easy to adjust, and you can update the database off-line or on-line, as individual records or all in one swoop.

Configuration of Web Pages

There's a host of options for configuring web pages. MiniVend has a complete tag language with over 80 different functions. You can embed code in the pages and use conditional statements like **[if ...]** ***text*** **[else] else-*text* [/else][/if]**, allowing for insertion of text or HTML code based on some computed choice.

All MiniVend commands are embedded in the pages within square brackets ([ ]). MiniVend preprocesses the pages before turning them over to the web server, replacing the commands with the values they represent. The most basic commands are the data tags that embed information from the database in the web page, but there are also commands for looping, inserting text from external files, inserting total fields and a lot more.

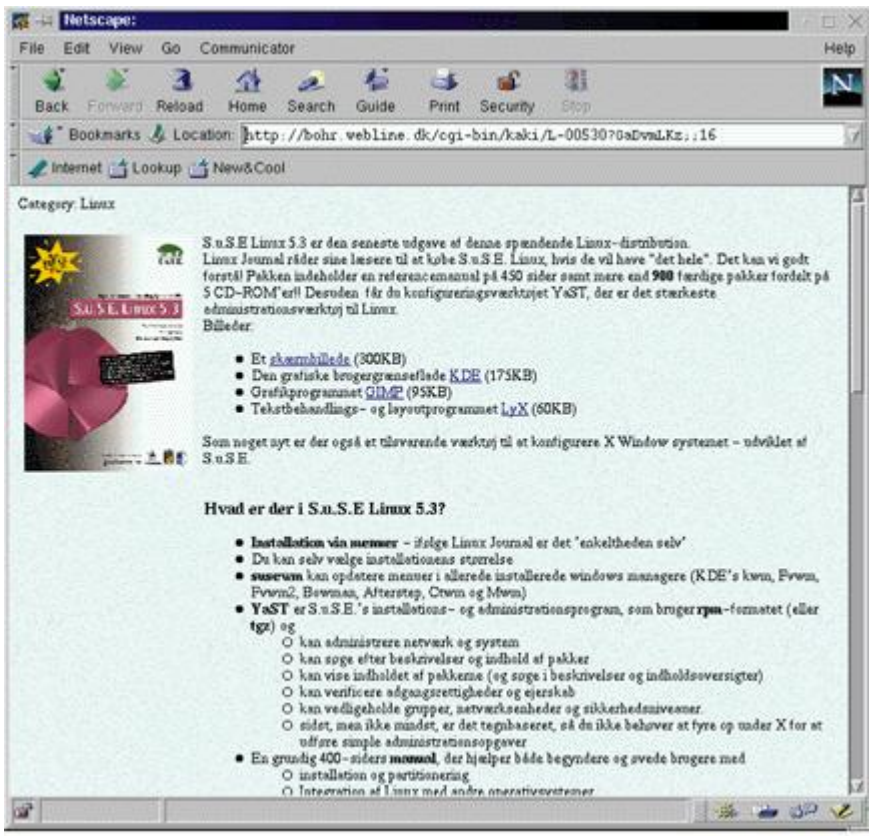


Figure 4. Example of a Configured Page

The example in Figure 4 was done with an external file holding all the text of the page. I named a field “webpage” and put it in the database, then inserted this in the template for the product page:

```
[include pages/products/[item-field webpage]]
```

which means I want to include a file at this point in the web page. The file should be found in the directory /pages/products/ and have the name that is taken from the web page field of that product.

As a developer, I was happy to learn you can even embed Perl code into the pages. At the same time, I have to agree with the documentation; it is a feature to be used rarely. Embedding Perl is like using a chain saw: you can get the job done quickly, but if used wrong, it can cost you an arm and a leg!

Administration

At the heart of MiniVend is the Products database. Normally, it is just an ASCII file with fields separated by tabs. The field names are given in the first row and MiniVend indexes the database with GDBM (GNU Database Manager) automatically. You can even hook up to a real SQL database through Perl's unique DBI interface, if you already have one.

You can update the database on the command line with the **offline** command, or if you just want to change one record, you can use the **update** command.

It is possible to import data from any source, provided the data can be formatted with the fields separated by tabs. If you want to adjust all the prices in the system, it is easy to do so with the **commonadjust** feature.

MiniVend records all sessions for future use. Obviously, some session data will become obsolete after a while, so it is a good idea to put an expire script in your crontab file:

```
44 4 * * * /home/httpd/mvend/bin/expireall -r
```

This will prevent your session databases from growing too large.

Conclusion

MiniVend is very impressive, almost awesome. This is software of a kind that other companies ask thousands of dollars (or euros) for, and MiniVend is free through the GPL. It is also a showpiece, demonstrating the versatility of Perl. A shopping cart system is not a small piece of code; MiniVend is a full-featured, powerful example.

MiniVend is a big system, not something you can cover in one session. It has scores of features I didn't even have time to try out yet. I've had a few problems with it, but I firmly believe that MiniVend is a system that can be used for any kind of electronic commerce. As I learn the features and study the code, I will become more familiar with it and can make it do all the things I want it to do.

If you're looking for a turn-key system, MiniVend is not it. On the other hand, if you are looking for a powerful, flexible and easy-to-use system, MiniVend is for you. The documentation is great, the source code is all there and if the support on the mailing list scales up a bit, there is nothing to fear.

Resources

Requirements

Kaare Rasmussen (kar@webline.dk) is a software engineer and developer, responsible for the software direction of a small Danish ISP known as Webline when work, family and other duties allow the time. Kaare has been working with almost all aspects of the IT industry for the last twenty years or so. He has written several books in Danish during the past five years—the latest one about using the Intranet with Linux.

Archive Index Issue Table of Contents

Advanced search

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

Introduction to Sybase, Part 1: Setting Up the Server

Jay Sissom

Issue #62, June 1999

Sybase comes to Linux—here's how it works.

Sybase has released their SQL Database product for Linux. This is a port of their full commercial database product—it is not crippled in any way. This series of articles will describe how to install and configure the server and how to set up and use clients. It would be impossible to teach everything about developing and maintaining client-server applications in a short series of articles, but these should get you started.

What is the Sybase SQL Server?

The Sybase SQL Server is an industrial-strength client-server database engine. It manages data and allows many clients to access this data efficiently and securely. The Sybase SQL Server allows you to concentrate on writing your application rather than on writing data access and security code.

Installing the Sybase SQL Server

Before installing the server, you must determine if your system can handle it. The server with all the documentation will require approximately 180MB of disk space plus space for your databases. A running server will take a minimum of 21MB of RAM. If you have many clients, you should give your server much more RAM; so for small development systems, a minimum of 32MB is required. For small-scale production servers, I would suggest a minimum of 64MB of RAM. As usual, the more RAM you give it, the faster it runs.

I have installed the server on numerous Red Hat 5.1 systems, as it requires the newer glibc libraries. If you have an older Linux installation that does not include the glibc libraries (Red Hat 4.2 and below), you should upgrade your system. If you have a non-Red Hat system, you will also need an RPM tool to install the files. The server is distributed as two RPM files. According to the

Sybase Linux web page (http://linux.sybase.com/), you can install the server only on Red Hat Linux 5.x and Caldera OpenLinux 1.3. In my opinion, you could install it on any Linux system—it would just be a bit more difficult than if you are using the Red Hat or Caldera version.

The web URL to access these files is http://linux.sybase.com/. This site will point you to one of the sites that offers the files for download. Remember this site—it contains important information you will need when using your SQL Server. After entering your registration information, you will be able to download two RPM files: one containing the Sybase SQL Server, the other with documentation in HTML and PostScript. This is approximately 40MB of data, so if you have a slow Internet connection, be prepared to wait a while. Fortunately, installing the software will take much less time than downloading it.

To install the server, log in to your Linux system as root and type the following command:

```
rpm -i sybase-ase-11.0.3.3-1.i386.rpm
```

You will be asked to read the license agreement; then it will install the Sybase server to the /opt/sybase directory. On my system, this is a problem because I usually don't have enough space on the partition to hold /opt/sybase, so I make a link that redirects /opt to /usr/local before installing. This works because it doesn't matter where the software is installed—it will work from any directory. From now on, I'll assume you have installed the server in its default location.

During the installation, you will be asked to create a group named **sybase** and a user named **sybase**. You will use the **sybase** user to perform maintenance on the database server. All users who need access to the Sybase server should be members of the group named **sybase**.

Now install the documentation files by running the following command:

```
rpm -i sybase-doc-11.0.3.3-1.i386.rpm
```

One thing the installation does not do is set the owner of the files correctly. While you are still logged in as root, issue the following command:

```
chown -R sybase:sybase /opt/sybase
```

Configuring the Sybase Server

Now that all the files are installed on your system, it is time to configure an SQL server and a backup server. A backup server is used to back up data in the SQL server while the SQL server is running. It guarantees that when restored, your database will have the proper integrity. Copying your database files from the

operating system will not guarantee that your database tables will be restored properly.

A single host can have multiple SQL servers if necessary. I would not recommend doing so, but it is an option. For our example, we will configure a single SQL server and a single backup server on our host.

Log in as the user **sybase** using the password you set when installing the SQL server. Since it is your first login as the **sybase** user, the login script will ask if you would like to run **sybinit**. The sybinit program is used to configure new and existing Sybase servers and is located in the /opt/sybase/install directory.

If you installed the documentation, you can follow along with the installation by using your browser and accessing the file /opt/sybase/doc/howto/howto-ase-quickstart.html. There isn't room in the magazine to display all the screens you will see. These screens are documented in the Quickstart Guide that comes with the server.

The first option you should pick is option 3, “Configure a Server product”. Configure the SQL Server first by selecting option 1. Since this is a new server, select option 1 again. Each server should have a unique name. I recommend naming the server based on its function. For example, a production decision support server could be named dss_prod. A development accounting server could be named acctng_dev. For this example, name the server linux_dev. When you finish filling in the data for a screen, press **ctrl-A** to save your data. Press **ctrl-A** now. At this point, nine more steps are required to configure this SQL server.

Select option 1. The interfaces file tells Sybase products where servers are located. Each server will listen on a specific port on its host. Just like SMTP mail, TELNET and web services, Sybase servers need a unique port to allow network connections. The interface file will hold the server name, host name and port number for each sybase server on your network. Select option 3 to add the port information for the linux_dev server. The sybinit program will automatically fill in the host name for your host. You should specify the TCP/IP port your server will listen on. For this example, select option 2, then specify 2360 as the port number. Any unused port will work. Press **ctrl-A** to save this entry in the interfaces file. After you confirm that everything is correct, press **ctrl-A** again.

If you have multiple sybase servers that you access, you can add information for each server into your interfaces file.

Before continuing, a little background information will be helpful. Database tables will hold the data for your application. For example, an accounts payable

application would have an invoice table, a vendor table and a payment table. A database can hold many tables. A Sybase server will have multiple databases. An accounting application might have general ledger, accounts payable and accounts receivable databases. When a server is installed, it will have four databases:

- **master**: The master database holds configuration data for the entire server.
- **model**: The model database is the basis for all new databases created on the server.
- **sybsystemprocs**: The sybsystemprocs database holds stored procedures used to maintain the server.
- **tempdb**: The tempdb database is a temporary workspace used when processing queries.

The sybase server manages disk space in devices. A device is a pre-allocated file of a specific size. A device file can be up to 2GB. A single server can have many devices. Databases are created on devices.

Select option 2. The master device holds the master database, the model database and the tempdb database. Its default size is 21MB. You can move the location of this file if you wish. If you have enough disk space, you can leave it in its current location. Press **ctrl-A** to save this screen. The sybinit program will give you a warning about the file name you selected. This is normal; on Linux, it will always give this warning. On other UNIX operating systems, the Sybase server devices should be raw partitions on the disk. This isn't possible on the current version of Linux, so we have to put our devices in operating system files.

Select option 3. The sybsystemprocs database contains stored procedures used to maintain the server. This database can also be used to store any procedures you write for server maintenance. I recommend you double the size of this database, so that you can add additional stored procedures to your server. When you do this, you have to select option 5 to increase the size of the device before you select option 1 to increase the size of the database. Put 32MB for options 5 and 1. You can leave the rest of the options as they are, unless you would like to place the device file in a different location. Press **ctrl-A** to save this screen.

Select option 4. As the server runs, it writes error messages to a text log file. This screen lets you set the location and name of this file. I recommend you give this file the same name as the database server. Change option 1 to /opt/sybase/install/linux_dev.log. Press **ctrl-A** to save this screen.

Select option 5. The database server needs to know the name of its backup server. I always give the backup server the same name as the database server, with _bs on the end. Change option 1 to linux_dev_bs. Press **ctrl-A** to save this screen.

Select option 6. The server can use many languages. I have never used anything other than us_english, so I can't tell you what will happen when choosing another language. For our example, just press **ctrl-A** to save the default for this screen.

Select option 7. You can configure which character set to use when communicating between a client and the server. Each client will tell the server which character set it should use. If you do not know for sure that you need another character set, you should accept the defaults for this screen. Press **ctrl-A** to save this screen.

Select option 8. Here you can specify which order to use when sorting data. By default, the server uses a binary order when sorting. This is the fastest sort method; however, when sorting words that are upper and lower case, the server uses the ASCII character set to sort so that uppercase letters are sorted before lowercase letters. Change the sort order to the dictionary sort order so that words are sorted properly regardless of case. Press **ctrl-A**.

Select option 9. If you wish, you can have the server maintain auditing information about users. For our example, we won't install auditing. Press **ctrl-A** to configure the server to not maintain auditing records.

We have now told the sybinit program everything it needs to know to configure your SQL server. Press **ctrl-A** to save your configuration. The sybinit program will now configure your SQL server. It will warn you about the master device file again, but it will create the devices and prepare the server for use. Your new SQL database server is now running on your system.

Press **ctrl-A** to go to the previous screen. The next step is to configure a new backup server. Select option 2, then option 1 to configure a new backup server. The name of the backup server should be linux_dev_bs. Press **ctrl-A** to save the backup server name.

I recommend changing the name of the backup server log file to /opt/sybase/install/linux_dev_bs.log. This server needs to be specified in the interfaces file also. It will listen on its own unique port. Select option 2. Select option 3 to add a new listener. As before, the host name has already been specified. Change option 2 to 2361. This will be the port for the backup server. Press **ctrl-A** to save this screen. Press **ctrl-A** again to write this entry in the interfaces file.

Everything else should be correct, so press **ctrl-A** to save this screen. The sybinit program will now configure the backup server and start it for you. Both the SQL server and the backup server should now be running on your host.

We are almost done, but there are three steps we need to finish. Press **ctrl-A** to go to the previous screen. Select option 4, “Configure an Open Client/Server Product”. These three products should be configured before you use them. To be honest, I don't know what configuring these products does because no options are available and they seem to work before configuring; however, it can't hurt to follow the directions. Select each of the three options one after the other to configure them. When you are finished, press **ctrl-A** to leave this screen. Press **ctrl-A** again to leave the sybinit program.

All of your entries are recorded in a log file. If you are configuring a production system, print this log file and keep it on hand in case you need to recreate the server that crashed at two in the morning.

Testing the Server

There is a basic client that comes with the Sybase SQL Server. This program is called **isql**. It is an interactive SQL program that allows you to enter SQL commands to the server and see the results. At the command line, type:

```
isql -Usa -Slinux_dev
```

The **-U** option tells the isql program which user name to use; **sa** is the system administrator account. It is similar to root on Linux. It has all rights in the server. The **-S** option specifies to which server to connect. In our case, the server name is linux_dev. The isql program will ask you for a password. Right after an installation, the sa user does not have a password, so just press enter. You can now enter SQL commands that will run on your SQL server.

The first command to run is a stored procedure that lets you change the password of a user. At the 1> prompt, type

```
sp_password null,'
```

sp_password is the name of the stored procedure. The sa user has no password, so we pass **null** as the first parameter. This parameter should be the password of the current user. The second parameter is the new password; put this password in quotes.

At the 2> prompt, type:

```
go
```

go tells the isql program to execute the command. If you make a mistake while typing a command before typing **go**, you can type **reset** to erase the command and try again.

All of the configuration information for the SQL server is stored in tables in the master database. Type the following:

```
1> select name from sysdatabases
2> go
```

This is an SQL command that queries the sysdatabases table. It will list the names of all the databases in your server. Almost all of the configuration information for the SQL server is stored in database tables. The documentation will give you more information on these tables.

To quit the isql program, type **quit** at the 1> prompt.

The SQL server comes with a script that will install an example database on your server. To install this database, type the following command at the $ prompt:

```
sql -Usa -i ~sybase/scripts/installpubs2 \
   -Slinux_dev
```

Type your new password when prompted. The script will create a database called pubs2, then create tables with data in them. You can now type queries like the following:

```
isql -Usa -Slinux_dev
Password:
1> use pubs2
2> go
1> select * from authors
2> go
au_id  au_lname    ...
---------- ------------------ ...
172-32-1176 White    ...
213-46-8915 Green    ...
...
1> quit
```

Shutting Down the Servers

Before you shut down your Linux system, you should shut down your Sybase servers. Do this using the isql program. Log in as the sa user to shut down the server. Once you are in isql, type **shutdown SYB_BACKUP** to shut down the backup server. **SYB_BACKUP** is the default name for a backup server. Then type **shutdown** to shut down the SQL server; this will remove both servers from memory. Now you can shut down your Linux system. If you don't shut down the servers properly, you could corrupt data. I recommend writing a script to perform this task automatically.

Starting the Servers

To start up the servers, you need to be logged in as the user **sybase**. Change to the install directory and type:

```
    ./startserver -f ./RUN_linux_dev
```

to start the SQL server and then

```
    ./startserver -f
   ./RUN_linux_dev_bs
```

to start the backup server. A startup script named /etc/rc.d/init.d/sybase is installed on your system. You can link this script to the proper places in your rc.d directories so the server will automatically start and stop when you start and stop your Linux system.

Conclusion

You have installed the SQL server and the backup server and you know how to start and stop it. There is still more to learn about the server. At the end of this article is a list of resources that can help you learn about your new Sybase SQL server. I recommend reading the PostScript documentation that comes with the server. If you don't want to print the hundreds of pages of documentation, you can use **ghostscript** to view them. For an easier way to view the documentation, go to http://sybooks.sybase.com/dynaweb and select Sybase Version 11.0.x Products. You can read all the documentation via Sybase's web site.

Next month's installment will be about writing database clients and installing the Sybase extension for Perl (**sybperl**) that will enable writing database clients in Perl.

Resources

Jay Sissom is responsible for the web-based front end to the financial decision support data at Indiana University. He has installed and supported Sybase databases on many operating systems and has written database clients for the Web using C, C++ and sybperl and for Windows using tools like Visual Basic and PowerBuilder. When he isn't programming, he enjoys amateur radio and playing his bass guitar and keyboards. If you have questions, you can contact him via e-mail at the address jsissom@indiana.edu.

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

LINUX JOURNAL

CORBA Program Development, Part 2

Mark J. Shacklette

Jeff Illian

Issue #62, June 1999

This month, the more advanced techniques of naming and event services are discussed.

In our last article, we introduced the concept of distributed programming with CORBA from a high-level point of view. In order to further flush out the CORBA infrastructure, we need to detail some of the standard services that the OMG (Object Management Group) has defined that should be supplied at least in part by most ORB vendors. Among these are the Trader Service, the Naming Service, the Event Service, the Interface Repository and the Implementation Repository.

The OMG has defined only the interface to each service while not attempting to provide an implementation. This means an OMG Service is actually nothing more than a CORBA interface written in IDL (Interface Definition Language). If a particular service is not available within a particular ORB or is not well-implemented, the developer always has the option of writing a custom implementation for the interface. In fact, if a vendor is truly CORBA-compliant, one vendor's implementation of a service can be used with another vendor's implementation of the ORB. This ability to mix and match CORBA-compliant implementations allows for flexible approaches to CORBA solutions. In this article, we will describe two of the most commonly provided OMG Services: the Naming Service and the Event Service. Our sample code is written using the feature-rich and GNU-licensed MICO CORBA implementation and demonstrates how to use both the Naming and Event Services in C++.

Last month, we introduced the concept of an IOR (Interoperable Object Reference), which we said was like a phone number or mailing address for the remote object. The client application can use the IOR to locate the remote object and establish communication. In that article, we handed the client

application the IOR by writing it to a file and passing the file to the server application at startup. In practice, this is an inconvenient way to design a system. One of the most common approaches to solving the problem of locating objects at runtime is to use the OMG Naming Service. The Naming Service is an interface to a database where an object's name is associated with its IOR.

In order to understand the Naming Service, it is often helpful to think in terms of the UNIX directory structure. The Naming Service is comprised of objects called naming contexts. A naming context can be thought of as a directory within a file system, ultimately deriving from a common root directory (the "root" context). Each name within a naming context must be unique. Since naming contexts are actually objects, a naming context can be registered with another naming context. In effect, this is analogous to creating a subdirectory within another directory in a file system. The hierarchical structure created by this method is called a naming graph. In order to simplify finding objects within a naming graph, the Naming Service allows objects to be referred to by compound names, which are similar to an absolute path name in UNIX.

The name under which an object is registered in the Naming Service is completely discretional and not required to even describe the actual object. In the Naming Service, the object's name is defined by a **NameComponent** object. These **NameComponent** objects are then stored in a particular naming context. The **NameComponent** object actually consists of two parts, an "identifier" and a "kind". The **NameComponent** is represented in IDL as:

```
struct NameComponent {
Istring id;
Istring kind;
};
```

Returning to the UNIX file system analogy, a UNIX file called Consumer.C would have an identifier of Consumer and a kind of C. In the same manner, an object may be stored in a naming context with an identifier of BusinessObject and a kind of java. The developer can thus use any naming standard he wishes when defining objects using the Naming Service.

In order for a CORBA client object to use the Naming Service to find other objects, it must know where to find the naming service. The preferred method of finding the Naming Service is to use the OMG method **resolve_initial_references**. Under most ORB solutions, resolve_initial_references will return the IOR of the "root naming context", or in effect, the root directory node.

In simplest terms, when a server application is launched, it registers or "binds" objects it wishes to expose with the Naming Service using compound names.

This is accomplished through the bind and rebind methods. The client application can then look up a particular object's IOR simply by resolving the object's compound name, which the client must know. The client application uses the resolve method to find an IOR from a given compound name. Once the name has been resolved and the IOR obtained, the application can narrow (narrowing an object is CORBA terminology for downcasting) the object reference to resolve the actual object implementation; from that point on, the object can be used as usual. Later, our example demonstrates how you might use the Naming Service to register and locate object implementations.

Another service with an OMG-defined interface is the Event Service. The OMG Event Service specification provides for decoupled message transfer between CORBA objects. The decoupling of communication provided by the Event Service allows for flexibility in terms of communication modes and methods. Specifically, it allows one object (Supplier) the ability to send messages to another object (Consumer) that is interested in receiving those messages without having to know where the receiver is or even whether the receiver is listening. This decoupling provides several important benefits:

- Suppliers and Consumers do not have to physically handle the communication and do not need any specific knowledge of each other. They simply connect to the Event Service, which mediates their communication.
- Message passing between the Supplier and Consumer takes place asynchronously. Message delivery does not need to entail blocking (although a pull Consumer may choose to block if it wishes—see below).
- Event Channels can be set up to be either typed or untyped (not all ORB implementations support typed events).
- Event Channels will automatically buffer received events until a suitable Consumer expresses interest in the events. Note that this does not imply either persistence or store and forward capabilities. Generally, an independent queue in the Event Channel will be devoted to each Consumer. These internal queues are generally based on a LIFO (last-in first-out) basis, with older messages disposed when the buffer is full and new messages arrive, without a Consumer extracting the messages fast enough. Most ORBs will allow you to set the maximum queue length.
- Events can be confirmed and can have their delivery guaranteed, if the vendor has implemented this capability.
- Suppliers can choose to either push events onto the channel (**push**) or have the channel request events from them (**pull**). Similarly, a Consumer may request to either synchronously (pull) or asynchronously (**try_pull**) obtain events from the channel, or have the channel deliver events to them (push).

- A one-to-one correspondence between Suppliers and Consumers is not necessary. There can be multiple Suppliers connected to a single Consumer via the Event Service, as well as a single Supplier connected to one or more Consumers.

Two primary styles of interaction exist between Suppliers and Consumers and the Event Channel: Push and Pull.

Push Method

In the Push Method, a Supplier will connect to the Event Channel and initiate a push of an event onto the Event Channel whenever it is ready to do so. It is the Event Channel's responsibility to buffer those events until they are delivered to one or more interested Consumers. In the Push Model, it is the Supplier that initiates the flow of events to the Event Channel. When a Supplier wants to connect to an Event Channel, it needs an object within the Event Channel to "pretend" it is a Consumer. This allows the Supplier to simply deliver events to its "Consumer", when in reality, its Consumer is simply a proxy for the actual Consumer, which is outside the Event Channel. It is to this proxy "Consumer" that the Push Supplier pushes events. Thus, the Proxy object is not a real Consumer, but merely an object within the Event Channel that provides a delivery mechanism through which the Supplier can deliver messages.

A Push Consumer will likewise connect to a "proxy" object, a proxy that represents the Push Supplier. When the Event Channel has a message available, the Push Supplier proxy will deliver (push) the message to the actual Consumer object. The message path is from the actual Push Supplier, through its Proxy Push Consumer, to the Proxy Push Supplier and finally to the Push Consumer itself. There are other variations of this, as our later example will show.

Pull Method

In the Pull Method, the Event Channel will pull data from the Supplier. In the Pull Model, it is the Consumer that drives the delivery of messages. A Pull Supplier will connect to a Proxy Pull Consumer. Again, as far as the Pull Supplier is concerned, it can consider this proxy object as a real Consumer that will request events periodically. An interested Pull Consumer object will then connect on the other end of the Event Channel to a Proxy Pull Supplier. When a Pull Consumer is ready to receive an event, it will initiate either a pull or try_pull call on its Proxy Pull Supplier, which will in turn query the Proxy Pull Consumer connected to the actual Pull Supplier, to request another event be delivered. In this way, the Consumer drives the data, when it is ready to process another message. Some implementations of the Pull Method will allow the Proxy Pull Supplier to periodically pull events from the Supplier at regular intervals in an

attempt to keep a buffer full of events for consumers when they request delivery.

The nice thing about the Event Channel abstraction is a communication does not need to be either entirely Push Model or Pull Model. A Push Supplier may indirectly connect to one or more Pull Consumers, and several Pull Suppliers may connect to one or more Push Consumers. It is the Event Channel logic that allows such interrelationships disproportionality among objects. It is the application design that drives the decisions concerning suppliers, consumers and their numbers.

Regardless of the relationship among suppliers and consumers, to establish a connection and deliver events through the Event Channel the following five steps must be taken:

- The client (Supplier or Consumer) must bind to the Event Channel, which must already have been created by someone, perhaps the client.
- The client must get an Admin object from the Event Channel.
- The client must obtain a proxy object from the Admin object—a Consumer Proxy for a Supplier client and a Supplier Proxy for a Consumer client.
- Add the Supplier or Consumer to the event channel via a connect call.
- Transfer data between the client and the Event Channel via a push, pull or try_pull call.

When messages are delivered through the Event Channel, they can be either “typed” or “untyped”. Typed messages are those defined in an IDL which are type-checked at compile time. Untyped events, the most common, adhere to the standard Event Services interfaces and are packaged as type **CORBA::Any**, which is a wrapper around all known CORBA types. It is this “Any” type that is actually sent from a Supplier Object to a Consumer Object. The Supplier will construct an Any, and the Consumer, upon receipt of the message, will derive the true value from the Any wrapper. This allows for great flexibility in delivering messages, as a Supplier may pass a string first, a long value second and an array third, all through packaging the values into an Any. The example code shows how to create, embed and extract values from Any types.

Our example incorporates both the Naming Service lookup as well as an implementation of a Supplier and a Consumer interacting through the use of the Event Service. The Supplier implements the Push Supplier Model and the Consumer implements the Pull Consumer Model, thus illustrating that the models do not have to be all of one type. Listing 1 shows Consumer.C, and Listing 2 shows Supplier.C. These listings are available by anonymous download in the file ftp://ftp.linuxjournal.com/pub/lj/listings/issue62/3213.tgz.

The first step the Consumer must take is to find the root naming context. This is accomplished by calling **resolve_initial_references** and then narrowing the returned IOR. The resulting object is the root naming context we can then use to resolve our Event Service.

```
CORBA::Object_var nsobj =
orb->resolve_initial_references("NameService");
assert(! CORBA::is_nil(nsobj));
CosNaming::NamingContext_var context =
CosNaming::NamingContext::_narrow(nsobj);
assert(! CORBA::is_nil(context));
```

As we turn to the Event Service sections of the code, we notice that the first thing the Consumer does after obtaining the initial context from the Naming Service is resolve and narrow the EventChannelFactory.

```
CosNaming::Name name;
name.length(1);
name[0].id =
CORBA::string_dup("EventChannelFactory");
name[0].kind = CORBA::string_dup("factory");
CORBA::Object_var obj;
obj = context->resolve(name);
```

MICO uses the factory referenced above as a generic CORBA::Object to create a new Event Channel object by first narrowing the generic reference, then calling the factory's **create_eventchannel** function:

```
SimpleEventChannelAdmin::EventChannelFactory_var
        factory;
CosEventChannelAdmin::EventChannel_var event_channel;
factory =
SimpleEventChannelAdmin::EventChannelFactory::_narrow(obj);
event_channel = factory->create_eventchannel();
```

We then use the Naming Service to bind this newly created Event Channel object to the name **TestEventChannel** via the Naming Service's bind method. This is done so that the Supplier will be able to locate this particular Event Channel by the name **TestEventChannel** when needed.

```
name.length(1);
name[0].id =
CORBA::string_dup("TestEventChannel");
name[0].kind = CORBA::string_dup("");
context->bind(name,<\n>
CosEventChannelAdmin::EventChannel::
_duplicate(event_channel));
```

Once the Event Channel has been created and named, the Event Channel object (**event_channel**) is used to obtain a reference to a ConsumerAdmin object through the **for_consumers** function. The ConsumerAdmin object provides the proxies for the Consumer clients of the Event Channel. It allows the Consumer to obtain the appropriate Supplier Proxy. In our case, we use the ConsumerAdmin object to provide us (a Pull Consumer) with a proxy Pull Supplier. This allows our Consumer object to act as if it were communicating directly with a Supplier that expects us to be “pulling” events from it. Of course, that's not actually the case. Our Supplier is really a Push Supplier that pushes

events onto the Event Channel. The proxies decouple the Consumer and Supplier objects and allow them to function as if they were directly connected, when in fact, their connection is indirect. Once we have the ConsumerAdmin, we use it to create our Push Consumer proxy:

```
CosEventChannelAdmin::ConsumerAdmin_var
Consumer_admin;
Consumer_admin = event_channel->for_consumers();
...
CosEventChannelAdmin::ProxyPullSupplier_var
proxy_Supplier;
proxy_Supplier =
Consumer_admin->obtain_pull_Supplier();
```

Once the Consumer has obtained a reference to its Supplier Proxy, it then notifies the Event Channel of its interest in receiving events from it through a call to the Proxy's **connect_pull_Consumer** method. An implementation of the Event Service's Pull Consumer interface is passed into the proxy_Supplier to make the connection.

```
proxy_Supplier->connect_pull_Consumer
(CosEventComm::PullConsumer::_duplicate(Consumer));
```

Once connected, calls can be made on the Proxy Pull Supplier's pull or try_pull functions. The **PullSupplier** interface is:

```
interface PullSupplier
  {
    any pull() raises(Disconnected);
    any try_pull(out boolean has_event)
         raises(Disconnected);
    void disconnect_pull_Supplier();
  };
```

In our case, we have the Consumer spawn a worker thread, and we pass the Pull Supplier Proxy reference to that thread, the one that actually makes the try_pull call. The try_pull call is an asynchronous polling mechanism allowing the Consumer to contact the Event Channel and “check for mail”. If there is a message in the Event Channel, that message will be returned as a **CORBA::Any** value, and the try_pull's **CORBA::Boolean** flag **has_event** will be set to true. The try_pull call is thus made from within the thread's “start” function in this way:

```
CORBA::Any* anyval;
CORBA::Boolean has_event = 0;
anyval = proxy_Supplier->try_pull(has_event);
```

If no event is waiting, the **has_event** flag is set to false and no value is returned; but the call does not block (as the pull function does), so it returns to the client immediately. This allows the client to continue doing other work while periodically checking to see if a new event message is waiting in the Event Channel's queue.

Once the **has_event** value is true and an Any value is retrieved, the Consumer must decide first what type it is, then extract that value from the Any wrapper in order to use it. The code to do that uses the Any's overloaded >>= operator. This strange-looking beast will attempt to extract the Any into the destination

type. If the type contained in the Any is compatible with the destination type, the value is extracted from the Any; if not, null is returned. The usual way to check for the value is to do something like the following:

```
if( *anyval >>= shortval )
  {
    cerr << "Consumer: thread pulled short:
         " << shortval << endl;
  }
  else if( *anyval >>= doubleval )
  {
    cerr << setiosflags(ios::fixed);
    cerr << "Consumer:
         thread pulled double: " << doubleval << endl;
  }
```

In our case, when we extract the correct type from the Any, we print it out and immediately begin checking again for events through our try_pull call.

Our Supplier implementation is a bit simpler. After binding to the ORB, it creates an implementation of a class that implements the CORBA PushSupplier IDL:

```
class PushSupplierImpl :
virtual public CosEventComm::PushSupplier_skel
{
public:
PushSupplierImpl() { }
void disconnect_push_Supplier();
};
...
  PushSupplierImpl * Supplier =
  new PushSupplierImpl();
```

This class implements the IDL PushSupplier interface, which has only a single function to implement: **disconnect_push_Supplier**. The implementation object, PushSupplierImpl * Supplier, will be used later to connect to the Event Channel and register our interest in supplying events to the Channel.

Just as the Consumer started by finding the root Naming Context, our Supplier begins by calling **resolve_initial_references**. Using the IOR returned by resolve_initial_references, the Supplier can then narrow to the naming context object.

```
CORBA::Object_var nsobj =
orb->resolve_initial_references("NameService");
assert(! CORBA::is_nil(nsobj));
cerr << "Supplier: successful call to \
resolve_initial_references()" << endl;
CosNaming::NamingContext_var context =
CosNaming::NamingContext::_narrow(nsobj);
assert(! CORBA::is_nil(context));
```

Once the name is resolved and narrowed, the Supplier attempts to retrieve a SupplierAdmin object through a call to the event channel's **for_suppliers** function.

```
CosNaming::Name name;
name.length(1);
name[0].id = CORBA::string_dup("TestEventChannel");
name[0].kind = CORBA::string_dup("");
CORBA::Object_var obj;
...
obj = context->resolve(name);
...
CosEventChannelAdmin::EventChannel_var
event_channel;
CosEventChannelAdmin::SupplierAdmin_var
Supplier_admin;
...
event_channel =
CosEventChannelAdmin::EventChannel::_narrow(obj);
Supplier_admin = event_channel->for_suppliers();
```

Once the SupplierAdmin object is retrieved, its **obtain_push_Consumer** function is called in order for the Supplier to obtain a Proxy PushConsumer with which to communicate.

```
CosEventChannelAdmin::ProxyPushConsumer_var
proxy_Consumer;
...
proxy_Consumer =
Supplier_admin->obtain_push_Consumer();
```

Once a proxy is obtained, we then need to connect the Supplier to the proxy through this call:

```
proxy_Consumer->connect_push_Supplier(
CosEventComm::PushSupplier::_duplicate(Supplier));
```

This call registers our interest in providing the Event Channel with events. The IDL interface for the PushConsumer (the implementation of which **ProxyPushConsumer** inherits) is:

```
interface PushConsumer
{
void push(in any data) raises(Disconnected);
void disconnect_push_Consumer();
};
```

Once a proxy push Consumer has been obtained, calls may be made on its push function, passing in a CORBA::Any value. This is done quite simply:

```
CORBA::Any any;
any <<=(CORBA::ULong) 555555555;
proxy_Consumer->push(any);
```

At this point, the Any value is delivered to the Event Channel, which is responsible for making that event message available to the try_pull calls of the Consumer, described above. Thus, we have come full circle in our discussion of the Supplier/Consumer roles in interacting with the Event Service.

Our example was built using the egcs 1.1b C++ compiler and MICO 2.2.1. In order to build and run the example, once you have unpacked the tar file, you simply need to update the variable **MICO_BASEDIR** in the Makefile to point to your base Mico installation, then type **make**. This will build both the Supplier and Consumer. To run the application, we've provided a simple script that

starts the rather lengthy MICO naming and event services for you automatically, then starts the Consumer (which creates the Event Channel), then the Supplier. To run the script, simply type **runit**. You will see the progress of the Supplier writing messages to the Event Channel, and the Consumer extracting them from the Event Channel; as it does so, it prints them out. Our Supplier will push, in succession, a long, a short, a double, a string, and finally another long (the number 13), which signals to the Consumer that it is finished. At that point, the Consumer thread exits and the applications are killed by the runit script.

Our next article will discuss an implementation of VisiBroker for Java that can be made available for development of clients and servers completely on Linux using Sun's JDK.

Resources

Home page for the Object Management Group: http://www.omg.org/

Introduction to CORBA: http://www.omg.org/news/begin.htm

The Free CORBA Page: http://adams.patriot.net/~tvalesky/freecorba.html

Java port for Linux: http://java.blackdown.org/

The CORBA FAQ: http://www.cerfnet.com/~mpcline/Corba-FAQ

Mark Shacklette is a principal with Pierce, Leverett & McIntyre in Chicago, specializing in distributed object technologies. He holds a degree from Harvard University and is currently finishing a Ph.D. in the Committee on Social Thought at the University of Chicago. He is an adjunct professor teaching UNIX at Oakton Community College. He lives in Des Plaines, Illinois with his wife, two sons and one cat. He can be reached at jmshackl@plm-inc.com.

Jeff Illian is a principal with Energy Mark, Inc. in Chicago, specializing in electric utility deregulation and distributed trading technologies. He holds a degree from Carnegie-Mellon University in Operations Research (Applied

Mathematics). He lives in Cary, Illinois with his wife, son and daughter. He can be reached at jeff.illian@energymark.com.

Archive Index Issue Table of Contents

Advanced search

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

Interview: Stephen Wockner of the TAB of Queensland

Bob Hepple

Issue #62, June 1999

A mission-critical application for 580 Linux computers.

The Totalisator Administration Board of Queensland (universally and affectionately known as the TAB, phew!) administers to the considerable betting needs of the Australian public in this sunny state of relaxation and fun. Virtually every pub (confusingly called a “hotel” here), club and high street has a high technology betting center for the Aussie punter (bettor). All to the tune of $1.3 billion Australian a year (about $900 million US)—making Queensland one of the biggest betting communities in the world, per capita.

From Bundaberg to Coolangatta, from Cairns to Woolongabba, the surfers and holiday makers on the Gold Coast and the farmers in the far outback are delighted to be ignorant of the fact that all of this technology is delivered and controlled by a massive distributed network of over 580 Linux computers running the TAB's own betting software. These Linux “branch controllers” are in every betting shop, agency, hotel and club to:

- Control the video displays of information to help the punters decide which horse to back.

- Place bets.
- Communicate to and from the head office mainframe computers in Brisbane.
- Let the punters access their bank funds through EFTPOS banking terminals (Electronic Funds Transfer at Point Of Sale).

In this interview, Stephen Wockner, Manager of Customer Systems and an 11-year veteran of TAB, takes us through the history of the TAB's use of Linux and some of the business imperatives that led to its adoption over more conventional systems at a time when such a decision was a very brave one.

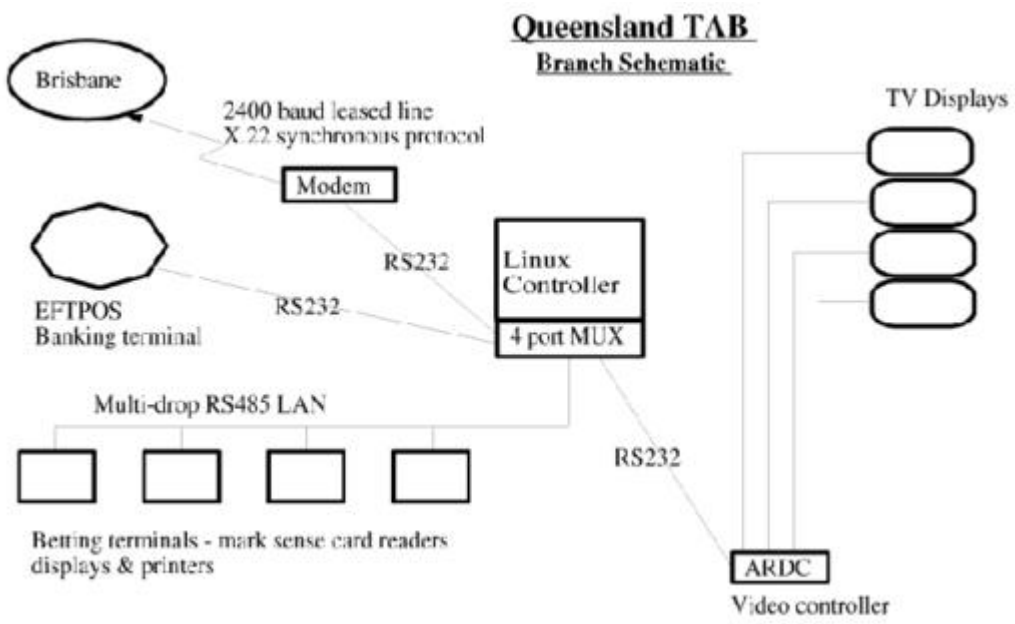


Figure 1. Queensland TAB Branch Schematic

Stephen (wockner@tabq.com.au) has been involved in computer systems development including VMS, Unix, NT and PC systems for over 17 years. He holds a master's degree in Technology Management. He enjoys both technical and business challenges, and Linux has certainly been able to provide them.

Bob: Please introduce the TAB for us.

Stephen: In the beginning, the Totalisator machine was invented in Australia in 1907—a mechanical device for calculating odds against bets placed and prize money. It's now in the PowerHouse Museum in Sydney. The TAB itself started business in about 1962 and is a statutory authority set up by the government with a board of directors appointed by the government—it will be an independent corporation next year. The product, mainly small bets on daily horse racing, is an inexpensive form of entertainment in pubs and clubs across the country and overseas where punters match their wits and knowledge of the horses against each other.

Out of every $100, the TAB returns $85 in prize money to the public with $6 going to taxes, $6 to the racing industry to fund the races and $3 to run the TAB; so, when you appreciate the scale of the organization across such a huge territory, it's a pretty lean and mean operation.

Bob: When did the TAB first implement Linux, and why?

Back in 1990, we had a system based on QNX2, a proprietary UNIX-like system with real-time extensions. This ran in every branch on [Intel] 386 machines with 4MB RAM, and controlled the entire shop, including running communications back to the head office over 2400 baud leased lines. It supported several video display devices to show punters the action and input devices for taking bets. These were pretty specialized “mark sense” card readers and printers.

It was a blow in 1994 when the manufacturers of QNX2 announced that it would be discontinued in favour of a new, better system called QNX4. The new system was to be POSIX compliant and full of new features to make it a true general purpose operating system, which unfortunately we did not need. QNX4 also came with a much bigger footprint—it would essentially require us to upgrade all of our betting outlets (then 250) to much larger hardware.

That led us to start looking around for alternatives—after all, if we have to upgrade the hardware, then we want to minimize the impact on the organization and the total cost as well as take advantage of newer software technology.

Figure 2. Mini-TAB in Use

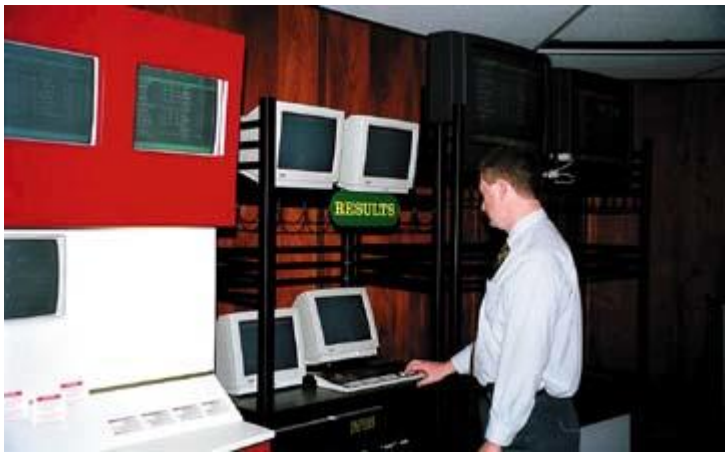


Figure 3. Installed Mini-TAB

Figure 4. Mini-TAB

Bob: What other systems did you investigate?

Stephen: We considered most of the operating systems available at the time, including DOS plus an add-on real-time multitasking manager, Windows 3.1, an early NT, a beta version of Windows 95 (Chicago), OS/2, Coherent UNIX, Xenix, Solaris, QNX4 and Linux 1.0.

Bob: What led you to choose Linux in the end?

Stephen: The process was a very objective one. We had no religious or other adherence to UNIX per se—we just had very specific business problems to address. In the end, these came down to a bottom line of cost minimization and the ability to do the job in over 500 locations, some very remote.

A number of the factors that went into the decision were:

- Functionality: it had to run on appropriate hardware of the day—486 with 16MB RAM in 1994, upgrading to Pentium today—and had to support the range of exotic devices we need. This included four TV displays via a black box controller, a four-port async port for displaying results and an RS485 multidrop LAN to control wagering terminals, typically four.
- Performance: support for particularly heavy interrupt-driven, heavily loaded, synchronous communications back to the head office using the X. 22 protocol over 2400bps HDLC (high-level data link control).
- Reliability: a system crash could easily lead to ugly scenes—punters can be pretty impatient when the race is only two minutes away. No, they are not happy to be told the system is rebooting. What you have to remember is that many punters watch the odds and place their bets at the last minute. Reliability and performance under pressure are key requirements for our computer system.
- Maintainability: many of our installations are not in town. They can be in the far north or outback areas with barkeeps who are not computer savvy or even permanent staff. The systems must be able to look after themselves with some remote management from here in Brisbane.
- Continuity: we had to be assured that the system would last a long time. Our systems have a life of five to seven years, and we learned the hard way that proprietary vendor support could not guarantee this. Also, we want to be sure that code written today will run on tomorrow's processors. So, as far as hardware is concerned, we use commodity level Intel PCs—the the fewer proprietary “improvements”, the better.
- Development environment: our philosophy is “insourcing”. Our applications are specialized and developed in-house or with third-party suppliers—all with source code available to us. A good development environment was important.
- Purchase and maintenance costs: licensing costs of the operating system add up quickly when 580 systems are involved. Some vendors cope better with this than others.

Bob: How did the different systems shape up?

Stephen: First of all, it quickly became obvious that Windows NT just could not cope with the the heavy communications loads and interrupt rates we would throw at it—even with top-end PCs. Windows 95 was pretty much the same. Running on a 486 with 16MB RAM could not even match the old QNX2 systems on a 386 with 4MB RAM. Reliability was another concern—the system had to keep running—the infamous “Blue Wall of Death” could become literal if punters are frustrated in placing their bets!

DOS with an add-on multitasker was just too difficult. We predicted continuity problems if we chose a third party supplier, and technical difficulty and delay if we wrote our own.

Coherent and Xenix were just too old—we had no confidence that support would continue. Similarly, OS/2 was technically nice but had a very limited future.

Solaris and other UNIX systems suffered from the same problem as QNX4—a big footprint, as well as unreasonable licensing rates. We were also uncertain about continued vendor support over our time scale of five to seven years.

In the end, Linux was the only system that could measure up on all fronts. The purchase cost was nice too, but by no means the most important factor—we would have been perfectly happy to pay a license fee (as we had with QNX2) for the right system. At that time in 1994, Linux was growing rapidly, and we could see that it had a great future.

Bob: How did the implementation go?

Stephen: We started work in early 1995 and made the first release of production systems in 1997. Some work was needed in getting the X.22 HDLC and RS485 networking going and porting the main betting application from the QNX2 system. Apart from having to fix some minor bugs, it went very smoothly.

As well as running our code and supporting our devices, Linux just keeps on going. So far, we have not had a riot at a betting shop because of a computer failure—in fact, our biggest reliability problem is power surges from the massive tropical storms we get here. Lightning can rip through even an e2fs disk and trash it. We don't know of many systems asked to operate in such a hostile and effectively unattended situation.

To cope with weather and other emergencies, we built a rescue partition into each system which can be booted remotely and which does just one thing—it rebuilds the master partition and restarts the machine. In this way, we have a timely repair for the most frequent service call—all without sending a technician 700 kilometers into the bush. We even have a facility (as yet unused) to reinstall the previous version of software in the event that we make a bad release.

Bob: Which version of Linux did you start with?

Stephen: Well, we're now on Slackware 3, kernel 2.0.14 but initially we used 1.0 and then 1.1—yes, don't look horrified, we actually ran production systems on the developer's versions and they were rock-solid—we needed the new

features. Essentially the access to source code allowed us to customise line drivers for our own purposes and to support ourselves. With the source in our hands, we could be confident we'd survive. In fact, we found ourselves fixing bugs in the 3c509 drivers.

Bob: And contributing them back, of course, as good netizens?

Stephen: Well, not always—the crazy thing was that, like many corporations at the time, the TAB distrusted the Internet. We are, after all, a major financial institution with high security requirements. We simply had no Internet access available at work—we used InfoMagic CD-ROM releases and had programmers hacking at home to download and upload additional code. Supporting Linux without the Internet was a major pain, so we may well have been consumers more than contributors in the early days. Today, of course, we have a full Internet connection—you can see us (and place bets on-line) at http://www.tabq.com.au/.

As far as contributing finished systems back to the community, we work with our partners—our contracts generally allow them to produce products and subsystems based on their work for us, so this code finds its way back into the system through them. Obviously, our top-level betting applications are too specialized to interest many people, and the issue of security is always present. We are mainly talking about subsystems.

A good example of this synthesis was the 20-port X.22 multiplexor card developed for us by Mosaic Pty Ltd and now marketed by them. It provides a very economical PC-based communications front end to our Unisys mainframes.

Bob: What about the end users? What advantages do they get from Linux?

Stephen: The advantage for them is they don't know it's there! It's reliable—it doesn't just fall over. It's simple for temporary staff or computer novices to learn and use—no mouse is involved, just a very simple set of cascading menus that are quick to learn and hard to cause trouble.

Bob: And the business of I.T.—how does that benefit?

Stephen: Linux truly helps us implement insourcing, which in turn allows us to maintain a tight loop for developing new features in our systems. The marketing people and top management think up new ways to bet, and we can support their ideas quickly. We're fairly fast on our feet, and time to market is a key issue.

All of this is done at a fraction of the cost of using a large vendor and becoming dependent on their proprietary lock-in features and escalating support costs. Most vendors are sales driven, after all. For example, if we buy a widget in year 0, how many of them are being sold in year 5 or 10? What motivation for continued support is there? We are confident that Linux will keep on pumping out the features we need, yet allow us to migrate our installed base at a rate we can control, not because some vendor needs income from a new version. So, the Open Source approach gives us control over continuity as well as the ability to exclude code bloat.

Efficient maintenance of such a huge system is also a key to keeping costs down. Sending a technician to many of the branches is a two-day affair with plane tickets and accommodation involved. Linux allows us to remotely monitor and maintain the systems—even install the next great version—all over a 2400 baud/s line!

Figure 5. The Linux Team at the TABQ

Figure 6. Linux Network Controller Glass House

Bob: Any problems or worries using Linux?

Stephen: Adopting it in 1995 was a big step for a corporation and was accompanied by the usual fears and objections of “where's the vendor to support us”. In fact, that fear has turned out to be a furphy (rumour)—the support we get off the Net is better than most large vendor's support, and our “insourcing” philosophy lets us modify the source if we get into a tight spot.

We have had to tool up to be independent and this is the downside. We needed to quickly develop a high level of skills and core competencies in-house and then make sure we keep the staff. We do this in the time-honoured fashion—we pay our engineers in the upper percentile and maintain a dual career path for them as they mature into either management or specialist engineers. Many companies don't provide this growth path and then wonder why their best people shuffle off.

Some of the wars over KDE vs GNOME and Linux Standards caused us to pause for thought—but by and large, it's a healthy battle and we would worry more if there was apathy over these issues. We are reassured that Linux is developed by a tight coherent community passionately concerned about the product.

Bob: What is the status quo?

Stephen: Currently, we have about 580 branch controllers installed with Linux systems, and we will continue to cycle out old 486 QNX systems in another 50 sites. We buy top-of-the-line commodity PCs—Pentium IIs now—and harden them on the developer's desks. As each batch of 50 or so comes in, the last lot goes out to the branches. We then work them to death—I mentioned the seven-year life cycle. The problem with that is we can't even donate our old stuff to worthy causes, as our old machines are really, really old and tired by the time we're through with them.

The ability to run Linux is obviously a prime acceptance criteria for all new equipment.

Bob: Any other applications for Linux?

Stephen: In another project, we have been putting Linux-based “mini- TABs” into the smaller outlets like pubs and clubs which don't justify the full system. These are about the size of a Coke machine, and we can make them at a reasonable cost—our counterparts in other states are not able to do this because of the high cost of mainframe support. Linux allows us to use a true client-server approach while preserving reliability to mainframe levels. We can also provide more information to the end user via Linux client-server, e.g., each screen supports 30x80 characters whereas the mainframe versions run at 24x30 (Teletext).

At the head office, we also use Linux as an authentication server for the entire network—no bet gets placed until the remote system is authenticated by Linux.

We also have about 20 boxes running QNX4 as a communications front end to the mainframe and talking to all the Linux boxes in the field. This is in place of

proprietary front ends costing $600,000 per 20 lines. Unfortunately, Linux doesn't have quite enough real-time oomph to do this yet, but as hardware speed increases and real-time features get added to Linux, we'll keep an eye on it.

By the way, the whole head office of mainframes and front-end processors and authentication servers is mirrored at a separate site in Queensland in case of catastrophe.

Finally, we have about 50 Linux machines in our development area which all run the X Window System and various desktops—we like Enlightenment, for example. The newer development environments are a very nice plus for us and really endorse our decision to use Linux—for example “Nedit”, which looks much like Virtual Studio and provides excellent productivity. Things like **gdb** are obviously well-used, but the nice new GUI front ends help keep us productive.

Bob: What about the Internet?

Stephen: We see the Internet as the preferred medium for the future of remote betting—in other words, phone betting. Obviously, it provides information in the most timely manner possible to our punters, which is the name of the game. By contrast, newspapers provide form and odds information which is over a day old, and radio and TV provide information only when they feel like it. Our customers presently use both phone or Internet, depending on which is more convenient for them. So that's a tip for punters—use the Internet for the most up-to-date information on racing.

We use Linux as Internet and security servers to provide digital certificates for our registered punters on the Internet. A Java applet on our betting page keeps information on their screen up to date—data is 30 seconds old or newer. You can't do better than that anywhere except by going into a TAB agency. The digital certificates issued by the Linux box have a special security feature—one we feel is essential in a financial environment—if a customer believes their certificate is compromised (e.g., stolen), they can call and cancel it within minutes. Not many (if any) certification authorities can provide this option.

Bob: And in the future?

Stephen: We have our telephone betting centres—call centres, really—with about 550 operators in Brisbane and 110 on the Gold Coast about an hour's drive south, all running on NT. We are at the testing stage of a new Linux version, which will use a digital sound system to store all calls to huge disc farms. This will replace the monstrous and antiquated tape recording system we have now and provide faster retrieval—we keep the recordings for 28 days

in case of a dispute with a punter (“I didn't say put $10 on `Neddy', I said `Freddy'”). This will be deployed late next year, and by the end of 1999, we will be running well over 1000 Linux machines.

The back-end mainframe is being converted to run on an NT platform. Subsequently, many of the applications written for NT are then ported to a Linux environment. In this conversion from NT to Linux, we'll be able to take advantage of the C++ code portability that we have always striven to maintain—we have a common set of libraries and take great advantage of reusable C++ code. For example, every system, whether NT, Linux or QNX, uses a common memory-based model of the entire betting system—almost an in-memory database. It's the same on all machines and is constantly kept updated.

On another front entirely, we are thinking of moving to satellite dial-up communications instead of leased line because Linux should be able to do this well. This could save us a big chunk of the $2 million we spend a year on fixed 2400bps lines and would let us use 19.2Kbps into the bargain.

Figure 7. Breakfast Creek Hotel in Queensland

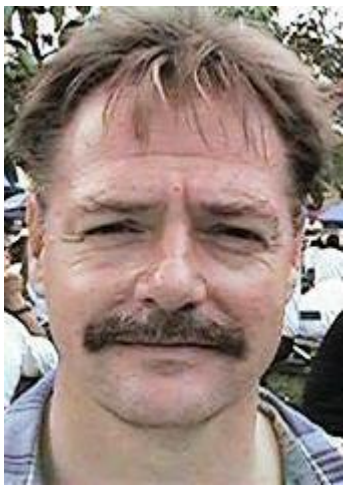
Bob: Any advice to others in the corporate and government I.T. world?

Stephen: Firstly, we have found through experience that it is much better to avoid the big bang approach to new system development as much as possible—where we have applied incremental improvements to individual components, we've had much greater success.

Linux suited our particular and special set of business problems in an objective evaluation and has produced the best and most cost-efficient, functional and reliable long-term solution. We were not fazed by the lack of a vendor standing behind the system even though it is, in every sense of the word, a mission-critical application. We feel that the FUD (fear, uncertainty and doubt) about vendor support is often a poorly disguised excuse for proprietary lockin. Of course, we have an insourcing and self-dependent philosophy which may not suit all entities.

Our cost savings using Linux goes far beyond the area of zero licensing fees into the less easily quantified areas of good old-fashioned reliability, serviceability, upgradability and continuity.

Bob: Thanks very much!



Bob Hepple is a 15-year UNIX and 5-year Linux veteran of HP, Sun, Unisys and Visa International in Singapore, Hong Kong and Australia. He originally learned to dislike the cold in the UK. Having recently moved to Brisbane in beautiful Queensland, he is resting between contracts. He can be reached at bhepple@bit.net.au.

Archive Index Issue Table of Contents

Advanced search

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

LINUX JOURNAL

Linux Clusters at NIST

Wayne J. Salamon

Alan Mink

Issue #62, June 1999

NIST is using Linux clusters for research, benchmarking them against supercomputers.

The National Institute of Standards and Technology (NIST) is experimenting with clusters based on commodity personal computers and local area network technology. The purpose of the experimental phase of the cluster project is to determine the viability of using commodity clusters for some of the NIST parallel computing workload. In addition to a Cray C90 running sequential codes, many parallel jobs are run on IBM SP2, SGI Origin 2000, SGI Onyx and Convex supercomputers.

Building clusters to run parallel jobs is done for several reasons. Low initial cost is one, although it is not clear how the long-term costs associated with clusters compare with those of traditionally packaged systems supported by manufacturers. Another reason is availability of systems. Having a slower system available without needing to wait is sometimes better than waiting a long time for access to a faster system. A third reason is that free implementations of the parallel virtual machine (PVM) and message passing interface (MPI) environments are available for Linux. Many NIST parallel applications rely on PVM or MPI.

Hardware Description

Figure 1 shows a diagram of the current cluster. Currently, there are 48 machines (nodes) in the cluster; 32 nodes have a 200 MHz Pentium-Pro microprocessor, and 16 nodes have a 333 MHz Pentium II microprocessor. The Pentium-Pro machines are built around the Intel VS440FX (Venus) motherboard, which uses a 33 MHz PCI bus and has four SIMM memory slots. The Pentium II machines use a motherboard based on the Intel 440LX chip set,

also with a 33 MHz PCI bus, and 3 DIMM slots for memory. All of the nodes are configured with a single 2.1GB hard disk supporting 512MB of swap space and the file systems. Sixteen nodes have 128MB of memory and 32 nodes have 256MB, for an aggregate total of 10GB of memory.

Figure 1. Cluster Diagram

Sixteen of the cluster machines are connected with both Fast-Ethernet and ATM. The remaining 32 machines are connected with Fast-Ethernet only. Therefore, the cluster can be configured with up to 48 Fast-Ethernet nodes or up to 16 ATM nodes, depending on the requirements of the job to be run. One other Pentium-Pro machine is used as an administrative front end to the cluster. This machine has both Fast-Ethernet and ATM interfaces.

The ATM interface cards are Efficient Networks ENI-155p with 512KB of memory onboard. These nodes are connected to a Fore ASX-1000 switch over OC-3 (155Mbps), using multi-mode fiber. For Fast-Ethernet, we use SMC EtherPower 10/100 and Intel EtherExpress 100+ cards connected to one Ethernet switch: an N-Base MegaSwitch 5000 with 60 Fast-Ethernet ports. The cabling to the switch is done with Category 5 twisted-pair cable. The Ethernet switch is connected to the intra-NIST network via a 100Mbps uplink to a router. Both the ATM and Fast-Ethernet interfaces are configured into different subnets, allowing us to monitor the network traffic independently. There is very little background traffic on the cluster subnets, since the nodes are used only for parallel application programs. Keeping non-computational network traffic to a minimum is important when evaluating the cluster network.

The cluster nodes have been augmented with the NIST-developed MultiKron performance measurement instrumentation (see Resources 6 and 7). The MultiKron PCI board is equipped with a MultiKron VLSI chip, a high precision clock and 16MB of memory. These features allow for precise interval measurement and storage of trace data with little perturbation. Another advantage of MultiKron is that the clocks on several boards (up to 16 at present) can be time-synchronized, allowing for tracing of events across the cluster with 25-nanosecond resolution. This type of measurement is important for precise tracing of network events.

Software Description

The 32 Fast-Ethernet-only cluster nodes are running Linux kernel version 2.0.29. We've found this version to be the most stable for our configuration. However, because support for the Intel EtherExpress 100+ is not in this kernel release, the device driver is built as a module. The 16 ATM/Ethernet nodes use Linux kernel 2.1.79, as this release is required for the ATM software version in use.

Development and support for the ATM software comes from the Linux-ATM project run by Werner Almesberger at the Swiss Federal Institute of Technology (EPFL). (See Resources 2.) We are currently running version 0.34 of this software, having started with version 0.26.

We installed Local Area Multicomputer (LAM) (an implementation of MPI) version 6.1 and Parallel Virtual Machine (PVM) version 3.10 on the cluster in order to run our benchmarks in addition to the NIST parallel jobs.

We also developed a device driver to allow for user-mode programs to control the performance counters present in the Intel Pentium-Pro and Pentium II processors. Two performance counters are present in the Intel Pentium-Pro architecture, along with a timestamp counter. Each counter can be configured to count one of several events, such as cache fetches and instruction executions. (See Resources 3.) The device driver is required because writing to the counter control registers (and the counters themselves) can be done only by the Linux kernel. User-mode programs can directly read counter values without incurring the overhead of a kernel system call to the device driver.

Another tool we use on the cluster is S-Check (see Resources 5), developed by our group. S-Check is a highly automated sensitivity analysis tool for programs. It predicts how refinements in parts of a program will affect performance by making local changes in code efficiencies and correlating these against overall program performance.

We have written many small test kernels to evaluate the performance of communication within the cluster. We have versions of the test kernels that communicate at the raw socket, IP and LAM/PVM library levels. These small kernels are useful in evaluating the overhead of the different communication software levels. By using the MultiKron toolkit, the kernels obtain very precise measurements of network performance.

Communication Benchmarks

The first step in assessing the performance of the cluster was to determine the performance of the cluster nodes in terms of memory bandwidth. Memory and bus bandwidth performance can limit the effective use of the network bandwidth.

We used a NIST benchmark, **memcopy**, to determine the main memory bandwidth of the cluster nodes. For buffer sizes greater than the cache size but smaller than main memory size, the size of the buffer transferred did not affect the transfer rate. On the 200 MHz Pentium-Pro machines, we measured a peak transfer rate of 86MBps (672Mbps). On the 333 MHz Pentium II machines, the measured rate was 168MBps. Both of these rates far exceeded the line speeds

of the ATM and Ethernet networks. Therefore, memory bandwidth is not a factor in utilizing the peak transfer rates of the network.

We measured the throughput and latency of the network using **netperf** (see Resources 4) and our own test kernel called **pingpong**. Using pingpong along with the MultiKron allowed a direct and precise measurement of the latency between cluster nodes. **netperf** was used to measure TCP and UDP performance, while variations of the pingpong program were used to measure the performance of the LAM, TCP, UDP and ATM socket levels.

Using the netperf stream benchmarks to measure throughput, we measured a peak rate of 133.88Mbps (86% of the OC-3 line rate) for TCP/IP over ATM. For TCP/IP over Ethernet, we measured 94.85Mbps (95% of the line rate). Both of these rates are near the maximum payload rate for the respective networks.

Measuring throughput with the pingpong program provided more insight into the network performance. While the netperf results tended to produce smooth curves, much more variability in the throughput was seen with pingpong as the message size increased. For messages below 16KB, Fast-Ethernet performed better than ATM when using TCP/IP. At this message size, Fast-Ethernet is near its maximum throughput, while ATM is not. With messages larger than 16KB, ATM throughput increases to surpass Fast-Ethernet.

While running the throughput tests, we noticed that the TCP/IP throughput drops dramatically when the message size is near 31KB. By using the MultiKron toolkit to probe the network stack in the Linux kernel, we were able to find the cause of the throughput drop. With the Linux 2.0.x kernels, transmission of the last message segment is delayed, even though the receiver window has opened to include room for the segment. We modified the kernel TCP software to prevent this delay, resulting in the elimination of the performance dip. (For details, see http://www.multikron.nist.gov/scalable/publications.html.)

Table 1. Network Latency (µs) for Message Size of 4 Bytes

To measure the latency of message transmission, we sampled the synchronized MultiKron clocks on the two cluster nodes involved in the data transfer. The latency is the time required for a minimum length message to be sent from one node to another. Table 1 gives the results of the measurements for different layers of the network stack. The values given are the one-way times from sender to receiver. Therefore, the TCP/IP measurement includes the device driver and switch time as well. Likewise, the device driver measurement includes the switch time.

Table 2. Ethernet Switch Latency (µs) for Various Message Sizes

The latency added by the ATM switch is greater than that of Fast-Ethernet for small messages. However, as message size grows, so does the latency added by the Fast-Ethernet switch, while the ATM switch latency stays constant. Table 2 shows the application layer latency when sending various-sized packets, using both the Fast-Ethernet switch and a crossover wire. As can be seen in the table, the latency added by the switch is between 123 and 131 microseconds. These latency values were consistent for several switches from different manufacturers. The cause is the buffering of each frame until it is completely received, rather than buffering only the header bytes, then overlapping the send and receive after the destination is determined from the frame header. (We have confirmed this with one switch manufacturer.) Although the latency is constant for each packet, it is easily hidden by pipelining for all but the first packet in a burst.

Application Performance

We have run several NIST applications on the cluster. Most of these applications are computation-bound, with little disk access. One exception is a speech processing job described below.

Our speech processing application is a batch job submitted piecemeal to each cluster node from a central server. This job was used to process over 100 hours of recorded speech. The processing involves analyzing the speech to produce a text translation. The job ran for nearly three weeks with little interruption on the cluster. The total CPU time used for the processing was over 42 million seconds across the 32 cluster nodes. Each piece of the job transfers 50MB of data from the central server via the Network File System (NFS) before starting the computation. Linux NFS has proven to be very stable. Overall, 6464 sub-jobs were run as part of the speech processing application, with 98.85% completing successfully.

Another NIST application run on the cluster, OA, predicts the optical absorption spectra of a variety of solids by considering the interaction of excitons. The bulk of the computation is based on a fast Fourier transform (FFT)/convolution method to calculate quantum mechanical integrals. The OA application was run on the cluster as well as Silicon Graphics (SGI) Origin and Onyx systems. Figure 2 shows the execution time of the OA application on the SGI and cluster systems. The best runtime occurred on the 8-node Origin, at 500 seconds, while the runtime on the 8-node ATM sub-cluster was 900 seconds. For the 16-node ATM sub-cluster, the runtime was only slightly better, showing that the application does not scale well beyond 8 nodes. The results show nearly a factor of two performance difference between the cluster and the Origin for this application, while the cost differential is more than a factor of ten. Running the job using the ATM network decreased the runtime by 30% compared to the

Fast-Ethernet network, where the runtime was 1300 seconds. This difference is due to the higher throughput obtainable over ATM.

Figure 2. Optical Absorption Performance

Figure 3. 3DMD Performance

The third NIST application, 3DMD, implements a three-dimensional matrix decomposition algorithm to solve elliptic partial differential equations. This application is considered “course-grained” because it generates large (100KB or more) messages at infrequent intervals. This application scales well as more nodes are added. Figure 3 shows the execution time of 3DMD on the SGI parallel computers and the cluster. With 16 cluster nodes, 3DMD ran faster than with the 8 Origin nodes (the maximum available on the Origin). For this application, there is a 10% performance difference between ATM and Fast-Ethernet, with ATM performing better.

Figure 4. NAS Parallel Benchmark Results

Figure 4 shows the execution time of the Numerical Aerodynamics Simulation (NAS) parallel benchmarks (see Resources 8). The NAS benchmarks are packaged as a suite of programs designed to measure the performance of parallel computers on computationally intensive aerophysics applications. The NAS suite is written in FORTRAN 77 using the MPI communication standard (distributed memory). The figure shows execution times for the NAS example problems when run on the 8-node SGI Origin, 8-node SGI Onyx and 8- and 16-node cluster using both Fast-Ethernet and ATM. The number in parentheses following the machine name gives the number of processors used for the problem run. The cluster competes well with the traditional parallel machines and ATM has an advantage over Fast-Ethernet for several of the benchmarks.

The second set of benchmarks we ran were the Stanford Parallel Applications for Shared Memory (SPLASH) (see Resources 9). This benchmark suite differs from NAS in that SPLASH utilizes shared memory as opposed to distributed memory. In order to run the SPLASH suite on the cluster, we used the TreadMarks (see Resources 10) Distributed Shared Memory (DSM) system. TreadMarks emulates DSM via the Network File System. Figure 5 shows the execution times for two of the SPLASH programs, Raytrace and LU. Raytrace consists of mostly computation, with very little communication, while LU spends a large percentage (nearly 35%) of its time communicating. The graph shows that for Raytrace, the cluster performs very well; however, for LU, cluster performance does not compare favorably with the parallel machines. This performance gap is due to the high communication overhead for small messages incurred on the cluster for the LU application.

Figure 5. SPLASH LU (L) & Raytrace (R)

The purpose of running the NAS, SPLASH and other benchmarks is to get a feel for the types of applications a cluster can run effectively. Also, for applications similar to the SPLASH LU, where communication time is the major factor in runtime, we need to delve deeper into the Linux network software and determine how network performance can be improved for these types of applications.

One other application for the cluster is the Distributed.NET project (see http://www.distributed.net/). During periods of low activity, the RC5 encryption software is run on the cluster nodes. Each node runs the software independently of the others, so we can participate with any number of nodes. We have run all 48 nodes (plus the front end) at times, with an aggregate key processing rate of over 32 million keys per second.

Conclusion

Linux has been beneficial in our research. The first device driver for the PCI MultiKron card was done on Linux and was the easiest to write. We use Linux to monitor the cluster, and the tools we develop are either written for Linux first or ported quickly from other UNIX environments. Experimenting with computing clusters would be more difficult with commercial operating systems because source code is generally not available. By having the ability to probe the operating system source code, we are able to accurately measure performance of the OS in addition to the performance of our applications.

Our experiments show that clusters compete very well with traditional parallel machines when running distributed memory applications, generally characterized by large messages. For shared memory applications, which tend to communicate with many small messages, the overhead of the network has a detrimental effect on the application performance. For both types of applications, tuning the network parameters can be of tremendous benefit in decreasing execution time.

The 333 MHz, 16-node Pentium-II cluster has been transferred into a production environment. This cluster will be made available to the entire NIST community and will be managed by the group that supports the traditional supercomputers. We believe Linux-based clusters will provide an effective environment for running many high-performance applications.

Resources

Acknowledgements

Wayne Salamon is a Computer Scientist within the Information Technology Laboratory at the National Institute of Standards and Technology in Gaithersburg, MD. He has worked on system software for PCs, UNIX workstations and IBM mainframes for the past 12 years. His current research interests are parallel computing and performance measurement. Wayne can be reached at wsalamon@nist.gov.

Alan Mink is project engineer of the Distributed Systems Technology project within the NIST Information Technology Laboratory. He holds a B.S. in Electrical Engineering from Rutgers University and an M.S. and Ph.D. in Electrical Engineering from the University of Maryland. His research interests include computer architecture and performance measurement. Alan can be reached at amink@nist.gov.

Archive Index Issue Table of Contents

Advanced search

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

LINUX JOURNAL

Sending Mail via the Web

Reuven M. Lerner

Issue #62, June 1999

Mr. Lerner continues his look at building a simple, integrated mail system that can be accessed using a web browser.

Last month, we looked at a simple CGI program (read-mail.pl) that allows us to view our e-mail from within a web browser. This program takes advantage of the fact that most e-mail today is delivered to a “post office” server, from which it is downloaded by a user's mail-reading program. Mail clients use the POP3 protocol to retrieve messages from a post office server, which means our program can retrieve a user's mail by connecting to a server and retrieving one or more messages.

read-mail.pl is good enough for basic purposes, in that it makes it possible to retrieve mail from any web browser in the world. However, while it makes it possible to read e-mail messages, it does not provide any mail-sending capabilities. True, many web browsers include such a capability—but many, such as Netscape Navigator, do not.

This month, then, we will take a look at how to send mail via the web. Between read-mail.pl (from last month) and send-mail.pl (in this article), we will have a simple, integrated mail system that allows users to perform all rudimentary tasks from any web browser.

Sending mail based on the contents of an HTML form was one of the first uses to which CGI programs were put, back in 1993 when CGI and HTML forms first arrived on the scene. As we will see, sending e-mail is not particularly difficult from within a CGI program. However, we will look at issues related to security as well as what we would need to do to turn these simple programs into a full-fledged Hotmail competitor.

Basic Mail Sending

Sending e-mail from within a program is normally quite straightforward, particularly if you are using Perl. You open a pipe to a mail-sending program, and send it the headers and data for the message you want to send. For instance, here is a simple program that sends a short “hello, world” message to my address, reuven@lerner.co.il:

```
#!/usr/bin/perl -w
use strict;
use diagnostics;
my $mailprog = '/usr/lib/sendmail';
my $recipient = 'reuven@lerner.co.il';
open (MAIL, "|$mailprog $recipient")
die "Cannot open $mailprog: $! ";
print MAIL "From: nobody\n";
print MAIL "To: $recipient\n";
print "\n";
print MAIL "Hello there!\n";
close MAIL;
```

There are several things to notice about this program. First of all, we set **$mailprog** to “/usr/lib/sendmail”, the default name and location of the mail transfer agent (MTA) on Linux systems. If your copy of sendmail is in another location or if you are using a program other than sendmail, you will need to change the value of **$mailprog**.

Similarly, mail is sent to a single address, what is defined in **$recipient**. We will discuss the issue of recipients later, when we look at the issue of program security. Keep in mind that restricting the number of recipients to which the program will send e-mail reduces the possibility that your program will be turned into a mail gateway by spammers or others interested in sending anonymous mail.

We open a connection to **$mailprog** using Perl's **open** function, which allows us to write to a program's standard input (STDIN) by treating the program name as a file name, and by prefacing the program's name with a | character. Anything we print or write to that file handle will be treated as if it were sent to the program's STDIN. Any output from the program is ignored.

Finally, notice how we insert a new line character between the final header and the message body. As with HTTP, SMTP (the “simple mail transfer protocol” used for most mail transfer on the Internet) requires a blank line between the header and any data. This allows the receiving program to identify which lines are headers and which are in the body.

Mail::Sendmail

Those of us who have been sending mail from within Perl programs were delighted when the **Mail::Sendmail** module, written by Milivoj Ivkovic, was

released to CPAN (the Comprehensive Perl Archive Network, based at http://www.cpan.org/). This module provides a portable method for sending mail from within a Perl program, but also provides a layer of abstraction between your program and the underlying mail system.

It is important to understand how the mail-sending mechanism works on your computer, particularly when it comes time to debug problems with sending or receiving e-mail. However, being able to send mail with three or four lines of Perl from a package maintained and updated independently of your program makes it possible to write shorter, more reliable programs. I have begun using Mail::Sendmail in all of my programs that send e-mail, and I suggest you do so as well, unless you have good reason to stick with the old system described above. One possible reason not to use Mail::Sendmail is if your program will be installed on a system without this package and on which you could not expect it to be installed. Given the ease with which packages can be downloaded and installed from CPAN, however, this should not deter you in a serious way.

The Mail::Sendmail module, like all other modules, must be imported at the top of any program that uses it with a **use** statement:

```
use Mail::Sendmail;
```

If you have not installed the module, or if it is not installed in one of the directories named in **@INC** (the path through which Perl searches when importing modules), Perl will fail with a fatal error.

After Mail::Sendmail is imported, sending a message becomes a two-step process. In the first stage, define a hash in which the keys are the headers and contents of the message. Specify the message body with the **Message** (or **Body** or **Text**) key. For example:

```
my %mail = (To => $recipient, From => $sender,
  Message => "Hello, there!");
```

You can then send the mail using the statement:

```
sendmail %mail;
```

The sendmail function is imported into the current name space automatically with the **use Mail::Sendmail** instruction. Mail::Sendmail defines many other functions as well, but none of these are imported into the default name space unless you explicitly request it.

As you can see, the above code is significantly shorter and easier to understand than what we did first. The fact that it is more portable and easier to maintain are nice additional benefits.

Moving to the Web

Now that we have seen how to send mail from within our program, we can concentrate on how to create a simple mail-sending facility from within a CGI program. Listing 1 shows an initial stab at send-mail.pl, which is a CGI wrapper around the above functionality.

Listing 1.

As you can see from the top of the program, send-mail.pl imports a large number of modules before it gets down to business. It uses **strict** and **diagnostics** to ensure our variables are lexicals (i.e., temporary variables defined with **my**), only hard references are used, and barewords are not considered subroutine calls. (Bareword is a Perl term for a word in which its use in a program is unclear. Originally, any such words were simply prohibited. Now that subroutines can be called without a leading &, barewords are interpreted as subroutines. This can confuse programmers and lead to buggy programs, so it is usually best to avoid them.)

Then, because this is a CGI program, we import the CGI.pm, a module which provides us with all the CGI functionality we could imagine, useful for receiving user input and sending output to a web browser. We also import **CGI::Carp**, which provides us with improved messages in the web server's error log. By importing the **fatalsToBrowser** symbol from CGI::Carp, we also ensure that fatal error messages are sent to the user's browser, as well as the error log. Normally, a fatal error in a CGI program results in an incomprehensible numeric error message on the user's browser. While the output from **fatalsToBrowser** might not seem much more useful or comprehensible to a non-programmer, it is not as scary as a set of numeric codes. Also, it makes the program much easier to debug than it would be otherwise.

Finally, we import Mail::Sendmail as described previously.

Other than retrieving three HTML form parameters (**sender**, **recipient** and **message**) and using them in the invocation of **Mail::Sendmail::sendmail**, this program contains little you have not seen before. We do want to ensure the mail is sent before reporting it has been, so we use **die** to exit with a fatal error; it will end the program after printing an error message to the user's browser and the error log.

We can determine if the mail was sent by checking the return value from the “sendmail” subroutine. If it returns **true**, we know the mail was sent. If it returns **false**, the program stopped before it was sent. Here is one simple way to accomplish this:

```
if (sendmail %mail)
{
# Print a message for success
}
else
{
die "Error sending mail: $Mail::Sendmail::error \n";
}
```

The variable **$Mail::Sendmail::error** (i.e., the variable **$error** inside of the package Mail::Sendmail) contains a detailed description of why the mail was not sent. Since the **sendmail** subroutine returns **true** when it succeeds and **false** when it fails, the above construct tells Perl, "try to send the mail contained in **%mail**--and if you cannot, exit and print a message describing why it failed."

If the mail is sent successfully, the user is returned a message indicating the program performed its task. It also prints the contents of the mail. Giving the user detailed feedback of this sort is always better than printing a simple "success" message, since the user might not be sure which e-mail message is being referenced.

Creating the Form

Now that we have a CGI program capable of sending mail, we need some way to invoke it. We could pass parameters as name-value pairs in the URL, but that is difficult and not very user friendly. We will thus send the name-value pairs using POST, which sends them to the program's standard input (STDIN). POST input to a program is generally sent from an HTML form. Here is a sample form that invokes send-mail.pl:

```
<HTML>
<Head>
<Title>Send e-mail!</Title>
</Head>
<Body>
<H1>Send e-mail!</H1>
<Form method="POST"
  action="/cgi-bin/send-mail.pl">
<P>Sender: <input type="text" name="sender"></P>
<P>Recipient: <input type="text"
  name="recipient"></P>
<P>Message:</P>
<textarea cols="60" rows="20"
  name="message"></textarea>
<P><input type="submit"></P>
</Form>
</Body>
</HTML>
```

This form has three elements, named **sender**, **recipient** and **message**. These are the same elements we retrieved with the **param** method in send-mail.pl. If you modify the names of the parameters in the HTML form, make sure to modify the program as well, or the form elements will not be picked up.

All HTML form elements are sent as name-value pairs in which the value is a text string. The CGI program receiving and interpreting the data does not know,

and furthermore does not have to know, whether the input field was a text field, a text area, a check box, a radio button or a pull-down menu.

Indeed, we can even substitute a hidden field—which does not appear on the web browser and cannot be changed by the user—for a text field, which comes in handy if we want to hard-code a value, such as that of the recipient. Simply replace the **recipient** line with

```
<input type="hidden" name="recipient"
value="reuven@lerner.co.il">
```

and all mail will be sent to my address.

Similarly, if you want to allow people to send mail to a number of addresses, but still restrict them somewhat, you can use a selection list:

```
<select name="recipient">
<option value="reuven@lerner.co.il">Reuven
<option value="eviltwin@lerner.co.il">Reuven's evil twin
<option value="info@linuxjournal.com.com">LJ editor
</select>
```

Changing our HTML form in any of these ways requires no changes to our CGI program. Once again, send-mail.pl expects to receive a name-value pair in which the name is **recipient** and the value is a valid e-mail address.

With the above form and CGI program in place, we should be able to send mail to any e-mail address on the Internet.

Preventing Spam

The problem with the above form is it truly allows anyone to send mail to any address on the Internet. Furthermore, it allows the sender to pretend to be any address on the Internet. This is precisely the sort of tool spammers love to exploit. If you were to put our original version of send-mail.pl on your site, you would eventually discover someone was using your server and bandwidth to send their spam.

Several possible ways can be used to prevent this. One is to remove the possibility of sending mail to users or domains outside of a selected list. For instance, we can define a hash where the keys are approved e-mail addresses:

```
my %approved_recipient = ('reuven@lerner.co.il' => 1,
  'info@linuxjournal.com.com' => 1);
```

Using a hash allows us to check the status of any e-mail address in a constant time interval, regardless of the number of addresses. If we were to use an array, for example, we would potentially have to search through the entire array before we could be sure of an address's status, meaning that the time

necessary to perform such a test would grow in proportion to the number of elements in the array.

We can thus check to see if an address is approved by inserting the following code:

```
if (!$approved_recipient{$recipient})
{
die "Unapproved address \"$recipient\": Mail" .
 " was not sent.\n";
}
```

A version of send-mail.pl with the above code can be found in Listing 2 in the archive file (ftp://ftp.linuxjournal.com/pub/lj/listings/issue62/3449.tgz).

We can similarly allow mail to be sent to any address within a particular domain by putting all of the approved domains inside an array:

```
my @approved_domains = ('lerner.co.il'
       'linuxjournal.com');
```

We then create a variable, **$match_found**, which defaults to 0:

```
my $match_found = 0;
```

$match_found will be set to 1 only if one of the approved domains matches the domain in **$recipient**. We check this with a short loop:

```
foreach my $domain (@approved_domains)
{
if ($recipient =~ m/$domain$/)
{
$match_found = 1;
last;
}
}
```

We use **last** to break out of the loop when we find a match, in order to save some time. If you know certain domains will receive mail more often than others, you should put them at the beginning of **@approved_domains**, since the earlier an item appears in that array, the sooner the match will be found.

We then send mail only if **$match_found** is true (i.e., non-zero). If **$match_found** is 0, we print an error message:

```
# If the domain was not approved
else
{
die "Mail was not sent: The recipient's domain " .
 "is not approved.\n";
}
```

The version of send-mail.pl in Listing 3 in the archive has these additions.

Checking for Errors

If we want our program to be robust, we must do more than check for security violations. We must check for input from the user that might not affect security, but might lead to bugs or other unpleasant surprises.

For instance, if we invoke send-mail.pl directly from a URL, for example

```
http://www.lerner.co.il/cgi-bin/send-mail.pl
```

the program will report that the mail was sent with a blank sender, recipient and message. This is bad for two reasons. First, no mail was sent, since necessary headers were not assigned any values, so the program is providing us with incorrect information. Second, we should never get to the point where blank data from the user is accepted as input for mail.

We can prevent this situation by ensuring send-mail is always invoked with POST, and that **$sender**, **$recipient** and **$message** are non-blank. If any of these is equivalent to the empty string, we exit prematurely from the program, telling the user each must have a non-blank value. Once again, using die is better in debugging environments than in production code, simply because of the style of error message it produces. There is no reason why you could not forward the user to an error message page, or print a nicely designed page describing what was missing, rather than simply dying.

Competing with Hotmail

Between send-mail.pl and read-mail.pl, we have created a small system to send and receive e-mail. Is this enough to compete with Hotmail creating our own small web-based mail service? The short answer is “no”, although the longer answer is that it is probably enough to suit most purposes.

Part of the problem is these two programs are run using CGI. While CGI is portable across platforms and languages, it is inherently slow, requiring the web server to create a new process each time the program is invoked. While this is more than adequate for lightly loaded machines, it quickly becomes a performance drain as the number of hits increases.

Each HTTP server has developed its own native interface that allows you to attach your program to the server process. Since Apache is free software, several such interfaces have been developed for it, including **mod_perl** and **mod_php**. The former allows you to write CGI-like programs in Perl, attaching them to the server process. This means your functionality becomes a subroutine within the server program, rather than an external program that must be invoked separately. The speed difference between a program running

under mod_perl and the same functionality in a CGI program is staggering and should convince just about any die-hard CGI user to switch to mod_perl.

A site wishing to compete with Hotmail would probably want to use mod_perl or a similar server-specific API in order to get the maximum performance out of its hardware.

Aside from performance, another issue is where the mail is to be stored. The programs we have discussed, read-mail.pl and send-mail.pl, expect the user's mail to be stored on a POP server elsewhere on the Internet. Hotmail and similar services have their own POP servers for incoming mail, as well as their own MTAs (usually sendmail, although other MTAs are apparently better for high-volume mail servers) running on their systems.

However, Hotmail will allow you to retrieve mail only from their own POP servers, while read-mail.pl will allow you to retrieve mail from any POP server, including one that would normally not have a web interface. Whether you restrict mail checking by users to your own system, a number of servers within your organization or anywhere else is up to you.

Finally, services such as Hotmail survive due to advertising, and one of the most popular ways to advertise is to add a short note to the bottom of each message indicating which web-based mail service was used to send it. We can easily do that by concatenating our own footer to the message the user sends with these instructions:

```
my $footer = "-\nBrought to you by ReuvenMail\n";
my $message = $query->param("message");
$message .= $footer;
my %mail = (To => $recipient, From => $sender,
  Message => $message);
```

Now everyone will know which mail service you were using when you sent mail from your web-based system. This functionality is included in the final version of the program, Listing 3 in the archive file.

Finally, Hotmail has millions of members, which means it is relying on more than a single computer running Linux for mail delivery. Operating a single system for sending and receiving mail is not nearly as hard as creating a large, scalable system. If you are interested in truly competing with Hotmail, you will need capital investment and a good knowledge of networking protocols, in addition to Linux, Apache and the above programs.

Conclusion

Sending mail from an HTML form is one of the oldest uses of the CGI standard. Many such programs have been created, although they have not always been

careful to restrict spam or other abuses. As we have seen, it is possible to get around most of these problems, but it is important to think about them before putting your system on-line.

By combining a simple mail-reading program with a simple mail-sending program, we can create a basic web-based mail service that is as open or closed as we desire. Perhaps these programs do not scale as well as Hotmail, but they do give us some insight into how that service (and similar ones) work, as well as what we might need to do with our own programs in order to make them as useful as possible.

Resources

Reuven M. Lerner is an Internet and Web consultant living in Haifa, Israel, who has been using the Web since early 1993. His book “Core Perl” will be published by Prentice-Hall in the spring. Reuven can be reached at reuven@lerner.co.il. The ATF home page including archives and discussion forums, is at http://www.lerner.co.il/atf/.

Archive Index Issue Table of Contents

Advanced search

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

Focus on Software

David A. Bandel

Issue #62, June 1999

nmap, Nessus, Saint and more.

Most of you are likely running the new Linux kernel. Whether from your favorite distribution or downloaded from the Web, this will seem like a small transition compared to what's coming. As I write, I am testing the new Caldera OpenLinux 2.2 beta. This distribution comes with not only a new kernel, but a new glibc as well—yes, a new system library. By publication time, I'm sure other distributions will be offering glibc-2.1, and I'll have heard much wailing and gnashing of teeth.

Several distributions, including Red Hat and Debian, have already included glibc distributions, based on glibc-2.0.7. As I remember, the glibc-2.0.x libraries were marked “experimental”, so if you experimented, oh well. You say Red Hat didn't mention the glibc library included in 5.*x* was experimental? I guess we all need to pay closer attention to those small details.

The new glibc-2.1 is different, and in some cases, incompatible. After installing the beta, I attempted to build **ssh** with no luck. A function call by needed ssh was missing. Also, my 128-bit encryption glibc Netscape binary wouldn't start—I had to install the libc5 binary of Netscape. These things should be fixed by the final release. This month, I'm still building with glibc-2.0.7.

nmap: http://www.insecure.org/nmap/index.html

nmap is a utility for mapping your network and open ports on the network. It is a very powerful, flexible, security auditing tool. While nmap has a number of legitimate uses, many options are available to perform “stealthy” probes of networks, something of questionable value. This tool will almost certainly become a favorite of “script kiddies” everywhere, so scanning your own network in advance to learn what they'll find will save you some headaches. At least, it will if you use the information to close/monitor any open holes that

were found. Several of nmap's options appear to be aimed at not triggering monitoring tools like **courtney** to report attacks. As a network and systems administrator, I consider probes of my systems and networks to be overtly hostile acts. At best, they will gain you a message to your zone technical contact; at worst, an entry in the hosts.deny file, sendmail access.db reject list and an ipchains drop packet entry. I know I'm not alone. Required libraries are libnsl and glibc.

Nessus: http://www.nessus.org/

Nessus is a highly configurable and very powerful security auditing tool. Like nmap, it will probe your network, looking for holes. Unlike nmap, Nessus requires a graphical interface, but provides a slightly more user-friendly report. You'll need to supply a bit more information to start it up, as it works in a server/client configuration. Nessus is also less subject to being “hijacked” by non-privileged users. If nmap is on your system in an accessible place, anyone can run it. Since the Nessus client must connect to a Nessus server and the server is password protected, ordinary users cannot make use of it as easily. You can make it even more secure by not leaving the server running. Required libraries are libX11, libXext, libXi, glibc, libdl, libgdk, libglib, libgmp2, libgtk, libm, libnsl and libresolv.

Saint: http://www.wwdsi.com/saint/

Saint is the reincarnation of SATAN. This particular tool will be comfortable to those who have used SATAN, but the license agreement bears reading. Based on the wording, I'd say their definition of “commercial” is significantly different from most definitions. The agreement appears to be more anti-litigation than restrictive of the use of the software. Still, it is a good tool. It requires the Perl 5 library and a web browser.

nettest: http://zorro.pangea.ca/~renec/nettest.php3

nettest is a fairly simple and extremely useful Perl script that will monitor any number of hosts for connectivity. It won't watch individual processes, but it will ping the host at designated intervals. If it notices a particular host has stopped responding (for whatever reason), it will take some action. That action may be no more than logging the event in syslog or e-mailing one or more addresses. If you know Perl, you can make it do even more. **nettest** can also be configured to take the same action when connectivity is restored. It requires the Perl library.

xfreecell: http://www2.giganet.net/~nakayama/

Freecell has been one of my favorite games for as long as I can remember. The addicting part of this game is that you know it's theoretically possible to win every game; however, I've yet to see anyone do it. While my average stays fairly high, occasionally I outsmart myself and just can't win—that doesn't stop me from trying. Fast animations give hours of fun. Required libraries are Xext, X11, stdc++, libm and glibc.

Ted: http://www.nllgg.nl/Ted/

Finally—a text editor that uses RTF (Rich Text Format) as its default format. This editor is a nice, very simple text processor. It will read ASCII text and RTF formatted files and write RTF, ASCII and HTML. I didn't test the HTML feature. I was mainly interested in the fact that it handles RTF, the one format any true word processor will understand. Spelling modules are available for Ted in English (American and British), Dutch, German, French, Spanish and Portuguese. Required libraries are glibc, libtiff, libjpeg, libpng and libgif.

David A. Bandel (dbandel@ix.netcom.com) is a Computer Network Consultant specializing in Linux. When he's not working, he can be found hacking his own system or enjoying the view of Seattle from an airplane.

Archive Index Issue Table of Contents

Advanced search

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

LINUX JOURNAL

Making Money in the Bazaar, Part 1

Bernie Thompson

Issue #62, June 1999

A look at the business models in use today and how they work.

Open Source is software which has been freed. It allows bits to be copied and reused endlessly. It allows inspection of the source code. It allows new innovations to be built upon old, without having to duplicate past efforts. It is free software with the emphasis on freedom.

This past year has seen an explosive rise in visibility for this curious market. The computer world at large has gained at least a limited understanding and respect for its workings. Much of this attention would have been unimaginable even a year or two ago.

During this time, Open Source has been put under heavy scrutiny. While certain technical benefits are undeniable, every analysis invariably confronts two simple, critical questions: “How does one make a living on free software?” and “Who is motivated to innovate?”

The strength of the answers to these questions will determine if Open Source will achieve its full potential for the greatest possible audience. It must be economically viable.

I will attempt to answer these two questions by surveying the field of current business models and analyzing their financial strength. I will also speculate on future innovations that may alter these dynamics.

Business Models

> Money rests on the axiom that every man is the owner of his mind and his effort ... Money permits you to obtain for your goods and your labor that which they are worth, but not more ... Money is your means of survival.

> —Ayn Rand, *Atlas Shrugged*

The obvious challenge of Open Source is that it may be copied freely, even if purchased initially. So a $10 Linux disc may legally be used to install one machine or a thousand machines. At first glance, it would seem no incentive exists to put effort into improving such a product. Because of this characteristic, Open Source is often equated with a kind of communism: a system that offers something for nothing and exploits the labor of others without rewarding them; in short, a system that is unsustainable because it causes people's self-interest to conflict with the greater good.

These concerns should not be dismissed out of hand, nor taken as factual. The truth is much more complicated. Central to these concerns is the lack of exclusive copyright protections. Copyright and patent laws are not inherently part of the free market; they are intended to create limited monopolies for the companies which own the rights. This is done to reward research and to encourage innovation.

Open Source is a voluntary system that waives exclusive ownership of software in exchange for other benefits. These benefits include wider adoption, faster collective innovation and a level competitive playing field. This makes for a frictionless, dynamic and highly competitive market without the very profitable “vendor lock-in” that is facilitated by traditional copyrighted software.

Despite the resulting competitiveness, several business models have proven to be profitable. These models leverage the unique new possibilities afforded by Open Source, in return for their sacrifice of certain copyrights.

What is still unclear is how these models will generate as much innovation and value as traditional software companies, given the handicap that a person's work can benefit his competition as much as it benefits himself. As we'll see, one of the surprising things about free software is where the innovations have originated.

In the following sections, I'll introduce the markets that are producing innovation and jobs today. These are the research, service and customization economies and the many business models that fall into these groups. Nearly all companies are hybrids of several different business models.

Table 1.

The Research Economy

Open-source corporations are important growth engines, but to date, they have built mostly upon the efforts of others. The bedrock of the market is the

thousands of individual students and moonlighting professionals who make small and large contributions every day.

These developers are not paid for their efforts. They begin a project with no promises or commitments. They work at their own pace, use their own judgment and set their own priorities. They are the university researchers and basement scientists of software, working together to make their contribution to the world.

Often, these developers are only learning or honing their craft, so many projects fail. Yet out of this soup of individual and group efforts rises some of the best software available today.

Through the Internet, these successful efforts can be instantly copied and put into use world-wide. They can be enhanced and customized by thousands of others. They can continue to evolve like an organism, adapting to new software and hardware architectures as the years go on.

The first and most unshakable answer to “Who will innovate?” is the students and moonlighters, motivated by their desire to learn and create and inspired by the energy and clarity of tackling new problems. The profit-oriented market may fail, but these software research activities will go on. Slowly, surely, they will continue to add to the body of free software available to the world.

Yet despite the best efforts of the students and moonlighters, their software has common flaws. Development goals are driven by the author's own needs, resulting in software “by developers, for developers”. The threshold at which a developer is satisfied with ease of use is much lower than for typical users. These are truly research projects, with all the beauties and warts that implies.

The Service Economy

In the cases where beauty has outweighed warts, a critical mass of technical and non-technical users has been built. Apache, Linux, Perl and many other programs have made this breakthrough to mass market utility. This expanded base of consumers has driven the need for many support services built around the software. These services add polish and value to the base provided by the initial project.

No one is required to pay for any services. Given only a Net connection, they can download what they need and figure everything out themselves. However, many consumers find their time more valuable and therefore seek services to make their lives easier. This is where “commercial” Open Source steps in to distribute the software, provide technical support and educate users.

Distribution

The Internet is great for downloading small software, but for larger products it is too slow. Also, finding the software you want among the jungle of projects on the Web can be difficult.

From this need rose companies like Red Hat, Caldera, SuSE, and Walnut Creek. On one convenient CD-ROM, you have an organized collection of the best available software applications. These companies are achieving significant revenue and earnings with these services.

Opportunities exist for further specialization. LinuxPPC has made a solid business out of focusing on the PowerPC market. A small company could take this further by taking one of the most popular (Dell, Linux, eMachines) PCs and producing a distribution that is tailored and tuned to that hardware. It could guarantee that all devices are recognized and install flawlessly. It could optimize every program to the particular processor used on that machine. You could imagine Intel co-marketing and co-developing with a software company to optimize for their latest chips.

Open Source can be a key in the drive towards mass specialization of computer products and services. As the overall size of the market increases, more opportunities will be created in these small niches and sub-niches. All of this is made possible by the full access to source code that free software provides.

Technical Support

When a single company owns exclusive rights to a software product, it is obvious where the most informed technical support comes from: if you buy a Microsoft product, you go to Microsoft for technical support.

The fact that Open Source does not have an exclusive support provider has repeatedly been portrayed as a weakness. This is a fundamentally flawed notion. Rather, Open Source allows a whole market of support providers to compete on a level playing field of equal access to the code.

Through this heightened competition, the level and quality of support is capable of rising above the best standards of today's closed software market.

Red Hat, LinuxCare and many other companies and individual consultants have stepped up to serve this market. Early in 1999, IBM recognized these extraordinary opportunities and announced Linux support and consulting services. The competition between these companies will become intense, and customers will be well-served.

If open-source support services can achieve their full potential, it will become a major selling point for corporate users and consumers. Innovations in providing these services will provide the foundation for many viable new businesses.

Education

O'Reilly & Associates has built a booming book publishing business which topped $40 million in 1998. More than half of this revenue was from books about free software topics.

As the market grows in size, more educational services will be needed. These are significant opportunities, since any educator, author or consultant can delve into the inner workings of the code to produce definitive training materials for a subject. By working on and teaching about specific areas, a valuable reputation can be created. (See Table 2, Linux Consultant Survey.) Several of the most successful consultants built their businesses by being a recognized world-wide expert in a particular technology.

The Customization Economy

The next step beyond servicing existing software is the creation of new applications to solve outstanding problems. This may be in the form of hardware devices that come preconfigured for a particular need. Or it may be through employees or consultants who configure and enhance software for particular needs.

In a world dominated by a single vendor, there are limits to the innovations a new product can provide because of high prices, too few features, too many features, logo requirements, etc. Many interesting new applications are suddenly possible when these shackles are removed. You just need freedom to customize.

Hardware Bundles

Hardware preloads and bundles are some of the most compelling uses of free software, because the cost of developing or enhancing free software for the machine can be included in the price of the hardware.

One example is the Cobalt Qube (http://www.cobaltnet.com/). This is a space-age blue 18.4x18.4x19.7cm server appliance running Linux on a RISC processor. It is a general purpose workgroup server for e-mail, Web, etc. Having full access to the Linux source code gave Cobalt the capability to fully customize the software for this uniquely simple but very powerful hardware platform. (See “Cobalt Qube Microserver” by Ralph Sims, October 1998.)

Another is the Snap! network storage server from Meridian Data (http://www.meridian-data.com/). It's a fixed-function server appliance that shares disk space on the network. It is built from custom hardware combined with open-source software. Consumers don't need to know it uses free software; they just need to know what it does. Customers expect the price of network storage to scale with the price of disk storage, so the hardware and software costs of using a proprietary software system could have greatly reduced the attractiveness of the product.

Obviously, one big advantage is having no per-device software royalty. This is particularly true for price-sensitive, high-volume products. In a few years, we may find dozens of companies embedding open-source operating systems and applications on millions of small, fixed-function hardware devices.

IT Professionals

Beyond hardware devices, there is a need to customize and adapt software applications to the exact business processes and needs of an organization.

This always requires some custom work. Most medium and large organizations have a crew of IT professionals whose job is to customize hardware and software to make the business run more smoothly. These professionals like to start with the most functional products possible and customize from there. This has meant proprietary software in most cases.

Recently, open-source software has achieved levels of functionality that match proprietary software in many cases and has the advantage of not being tied to one vendor for support or product updates.

Rather quickly, it may become cost-effective to customize free software, rather than pay for thousands of licenses of commercial software on which to build. This shift in the market will require a growing number of professionals who specialize in open-source software.

This is perhaps best reflected in the salaries of IT Professionals. A 1998 salary survey of 7189 professionals asked which operating system they primarily used. Of those who reported Linux as their primary OS, their salary was $61,027 US vs. an overall average of $60,991. Linux salaries had increased 16.5% from the previous year, representing the fastest salary increase of any system (source: Sans Institute).

In-house staff is not the only option. Again, because of the freedom to inspect and study the software down to the lowest levels, a competitive industry is able

to grow to serve whatever needs arise. The resulting alternative to in-house staff is a competitive market of independent consultants.

Consultants

When the cost of the software goes to zero, the value is in customizing for specific problems. Consultants already make their living providing these per-hour or per-project services. Open Source is not a sacrifice; it is an opportunity.

One example is comprehensive support. Most business want a single point of contact to take full responsibility for getting a project done. With closed source, contractors are at the mercy of bugs and limitations in the operating systems and applications they purchase. In effect, they cannot guarantee success. They do not have full control of the technology.

With Open Source, they have complete access to solve every problem, no matter what level or layer it occurs in. A small company with a skilled force of engineers can provide a level of comprehensive application-to-operating system service that only IBM or HP or Sun can provide today, and probably at much lower prices.

To get an understanding of the size and health of the Open Source consulting market, those registered in the Linux Consultants HOWTO were surveyed. They were asked the following questions:

1. How many consultants at your company are involved with Open Source work?
2. Approximately how much money did your company (or yourself, if independent) earn in 1998 on Open Source-related work? (Convert to US dollars)
3. In 1999, based on numbers from recent months, how much do you expect this to increase/decrease? (as a percentage)
4. In 1999, do you believe it is possible to make a living doing Open Source consulting work? (yes/no)

This is a very diverse group of VARs, integrators and consultants. Over 50% are from outside the U.S., where the cost of living may sometimes be lower. In most cases, open-source work is just a piece of the total business. While this is certainly not a scientifically rigorous study, it does give some flavor of the market.

Table 2.

A key point from the survey is the importance of being a “jack of all trades.” You must focus on serving the needs of the customer, including doing work on closed source. In 1998, the median earnings per consultant on Open Source alone were not enough to make a living, and only 12.7% of the consultants made more than the $61,027 salary of IT professionals mentioned above. Business has picked up dramatically in recent months, however. As a whole, the consultants were very bullish on the coming year.

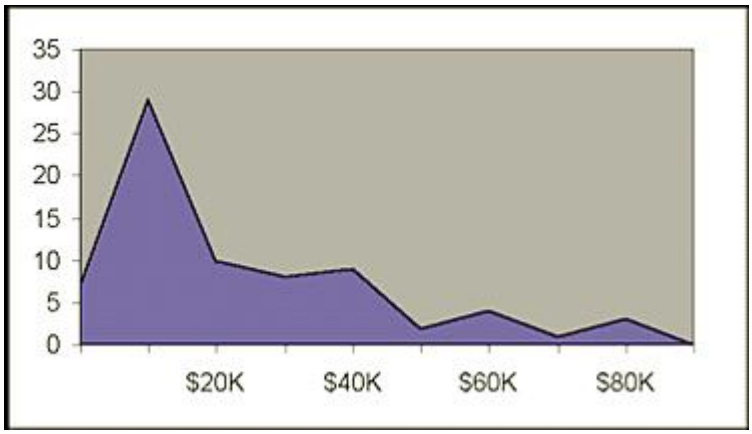


Figure 1. Survey: Distribution of Earnings

In the previous sections, we've covered the current business models that provide a living for employees, and innovations for consumers. There are certainly strengths, but the market is still tiny compared to traditional shrink-wrapped software. Young companies with new ideas are needed in order to grow the market.

Funding New Companies

Capital is the fuel for companies that will serve any new market. This money may come from the on-going operations of the business or from banks or investors. What is the current environment for getting this funding?

Venture capitalists, the investment partnerships that fund high-risk/high-return companies, are skeptical so far. Their analysis of these opportunities keeps coming back to a critical point: Open Source, by definition, eliminates the barriers of entry to a market. How can a company build a sustainable market advantage if their work can immediately be used by a competitor?

Table 3.

Given this limit on the upside, only a limited number of open-source companies have received funding. These companies have identified key factors to protect them from competitors. For Red Hat, it is a strong brand. For Sendmail, it is having an open/closed mix in their software product line. For a company like Cobalt Networks, it is combining closed hardware with open software. As this

market matures, more companies may achieve profitability and attract investment dollars for everyone.

Until then, companies must bootstrap themselves. Ironically, this is feasible because of those same low barriers to entry that scare off investors. An open-source company can build on the past efforts of others, meaning less capital is required to start the company.

Problems to be Solved

In summary, what are the problems that companies must solve in order to grow the market in new directions?

- The financial motivation for innovation must be stronger. Most of the current successful business models other than consultants make money off "secondary" services, rather than the software development itself.
- Open Source is still largely "by developers, for developers". To achieve mass market success, it must become more customer-driven and consumer-friendly.

Traditional software products harness the free market to solve these issues. Consumers pay to buy a software product if it meets their needs, which means it must be very polished. Successful products are profitable for the companies that create them. Unsuccessful products die off. Through these mechanisms, good developers make a living and consumers get good choices.

Open Source needs to create systems to provide these consumer checks and balances.

The Search for Solutions

The business models described throughout this article are by no means a comprehensive list. This is a young market we are only beginning to understand. It could yet defy the skeptics and evolve into something that serves customers better and is financially strong.

Part two of this series will explore one particular possibility in this universe of interesting but unproven ideas—a consumer co-op for software contracts. It uses the Web to let consumers commit funds up front to pay for the development of specific applications, feature enhancements or bug fixes critical to them. Resources are pooled, so each person pays only a small portion of the total cost. It is a system compatible with, and tailored to, Open Source. I will analyze this idea in detail, describe an attempt to create a web service which provides the necessary mechanisms and speculate how this system might affect the progress of the open-source market.

With this idea and many others, the open-source market is a fascinating mix of possibilities and dangers. In recent years, it has grown from thousands to millions of users. Several profitable companies are now serving the needs of these consumers.

The next few years will certainly see continued innovation from the open-source research community. From the business side, it remains to be seen whether the current momentum will continue or be struck down by market realities. It may very well depend on the innovations created by the upcoming generation of open-source entrepreneurs. It's a free market. May the best products win.

Bernie Thompson has worked in both “evil empires”: IBM and Microsoft. He also was published in the very first issue of *Linux Journal*. Please send any comments or requests to bernie@veriteam.com or see http://www.veriteam.com/.

Archive Index Issue Table of Contents

Advanced search

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

IP Bandwidth Management

Jamal Hadi Salim

Issue #62, June 1999

A look at the new traffic control code in the kernel and how it aids in bandwidth management.

The success of the Internet is attributed mainly to the simplicity and robustness of the protocol that ties it together, IP (Internet Protocol). Lately, everyone wants to run on IP. Major drivers include telephone companies wishing to replace traditional circuit-switched voice networks by carrying voice on IP networks, and multi-site corporations wanting to replace their dedicated connections with virtual private networks (VPNs) over shared IP networks.

However, IP suffers a small handicap. Unlike protocols such as ATM, it treats everyone equally. All data that goes through IP networks is equally forwarded on a *best-effort* basis. What if I was willing to pay $2 more a month so customers could get my web pages loaded about half a second faster? What if I was willing to pay a little more so I could have a coherent audio conversation, across the Internet, with someone across the Atlantic? In both instances, those particular willing-to-pay-more packets will have to be treated *more fairly* for this to work, thus, IPs equality for all fails. Now, the new big buzzword is here: *Quality of Service* (QoS), that is, trying to streamline IP to meet these new requirements. Although not a new concept per se, QoS has gained more momentum lately due to the interest of large corporations in using IP.

QoS means different things to different people. An order of a burger and fries at McDonald's is cheaper than at a fancy restaurant, where you are served a glass of water and lots of courtesy before your order arrives. You pay more at the restaurant for the *QoS*. Someone might argue that the QoS is better at McDonald's because you get served faster. Another analogy is the airline model, where the same plane has economy and first-class customers. In simplistic terms, one can define QoS as paying *more* to get *better* service. As such, it is a good way for the Net to be self-sustaining.

An implication embedded in this is that the socialist days of the Internet are over. Socially, the advent of IP-QoS is already being blamed for introducing a caste system on the Internet: the “bit-haves” and “bit-have-nots” are becoming reality.

The ability to divvy up bandwidth owned by a service provider for QoS is referred to as “bandwidth management”. Several techniques have been proposed and implemented over the years. The Internet Engineering Task Force (IETF) has in the past proposed integrated services, via RSVP, which is host driven. RSVP has failed to take off as a widely deployed standard, mainly due to scalability issues. Currently, the IETF is pushing a new solution known as differentiated services (diffserv), which gives more control of bandwidth management to network owners. This article will not delve into the details of the two techniques. The good news is that both are currently supported in Linux.

The unsung hero of the new 2.1.*x* Linux traffic control (TC) code (and much more) is Alexey Kuznetsov (kuznet@ms2.inr.ac.ru). Alexey invested a great deal of thought in the design in order to make it extremely flexible and extensible.

What I describe is just the tip of the iceberg of the possibilities presented by Linux traffic control, without going deep into detail. The intended scope is to show via a simple example how one could unleash the power of Linux traffic control.

Primitive Bandwidth Management

The TC features give ISPs the ability to manage (or carve) their bandwidth as they see fit. In the past, there have been other less-organized ways of doing this. The ISP could bandwidth-limit the customer's access rate by selling services based on interface capabilities, e.g., 28.8 vs. 56Kbps modems or 1 vs. 3Mbps xDSL modems.

Another more innovative but less ambitious (relative to TC) way of rate-limiting bandwidth is to use Alan Cox's shaper device. The shaper device is first attached to an already configured network device (e.g., Ethernet) using the **shapecfg** utility, which is also used to configure the shaper's speed. The next step is to use the **ifconfig** utility to configure the shaper to have the same IP address as the device to which it is attached. The final step is to map the packets to be treated by the shaper; this is known as *classification*. This is done using the common **route** command, pointing the route in which the packets are to be conditioned to the shaper. The advantage of the shaper is that it also runs on the 2.0.*x* kernels (included from 2.0.36 onwards and available as a patch for earlier kernels). The shaper's limited classification capabilities can be enhanced

in the 2.0.*x* kernels by using Mike McLagan's (mmclagan@linux.org) patch to allow routes to be specified by source/destination pairs. The new TC capabilities encompass shaping as well as a great deal more.

Another technique to enable bandwidth management is to use the multi-routing table capabilities implemented by Alexey Kuznetsov. Linux 2.2 has a new feature that allows a single Linux box to have multiple routing tables. In essence, one could have a special routing table for a higher-paying customer and redirect their traffic through higher-bandwidth or less congested devices (e.g., to a T3 rather than an ISDN line, both of which are headed the same way). Perhaps the best-kept secret in Linux bandwidth management is that the Apache web server has a bandwidth limiting module, **mod_throttle**, to rate limit individual users as defined in a config file. For details, see http://www.bigrock.com/~mlovell/throttle/.

2.1.*x* Traffic Control

The Linux TC conditions packets after the next hop has been decided, i.e., after the forwarding code has decided which interface the packet will go out on. This means that only outgoing packets are subjected to the TC. The TC consists of three building blocks.

The **queueing discipline** can be thought of as the traffic/data-packet manager for a device. It *encapsulates* within it the two other major TC components and controls how data flows through them. Only one such managing component can be attached to a device. Currently, a few device queueing disciplines are available to manage devices, including class-based queueing (CBQ), Priority and CSZ (Clark-Shenker-Zhang). An example configuration with CBQ will be shown later.

The **class(es)** are managed by the device queueing discipline. A class consists of rules for messaging data owned by that class. For example, all data packets in a class could be subjected to a rate limit of 1Mbps and allowed to overshoot up to 3Mbps between the hours of midnight and 6AM. Several queueing disciplines can be *attached* to classes, including FIFO (First-In-First-Out), RED (Random Early Detection), SFQ (Stochastic Fair Queueing) and Token Bucket. If no queueing discipline is attached to a device, basic FIFO is used. In the example shown later, no specific class queueing disciplines are attached, thus defaulting to simple FIFO. CBQ, CSZ and Priority can also be used for classes and allow for subclassing within a class. This shows how easily very complex scenarios using TC can be built. The queueing disciplines managing classes are referred to as class queueing disciplines. Generally, the class queueing discipline manages the data and queues for that class and can decide to delay, drop or reclassify the packets it manages.

Classifiers or filters describe packets and map them into classes managed by the queueing disciplines. These normally provide simple description languages to specify how to select packets and map them to classes. Currently, several filters (depending on your needs) are available in conjunction with TC, including the route-based classifier, the RSVP classifier (one for IPV4 and another for IPV6) and the u32 classifier. All of the firewalling filters can be used subject to their internal filtering tags. For example, **ipchains** could be used to classify packets.

The TC code resides in the kernel and the different blocks can be compiled in as modules or straight into the kernel. Communication and configuration of the kernel code or modules is achieved by the user-level program **tc** written by Alexey. The interaction is shown in Figure 1. The tc program can be downloaded from ftp://linux.wauug.org/pub/net/ip-routing/iproute2-current.tar.gz. You will need to patch it for glibc, if you are using a glibc-only system. The patches are available in the same directory. Note that the package also includes the **ip** and **rtmon** tools.

Figure 1. Configuration of Kernel Code with TC

TC Features

TC is very flexible: you decide what you want to configure as a service. An ISP that offers virtual servers with different QoS levels is a good example of the power of Linux traffic control. Note that similar services can be applied within an intranet. The traditional offer differentiator when ISPs sell virtual, web-server, hosting services is disk space. For $5 more a month, you could get an additional 100 megabytes of disk space for your hosted web server. Other ISPs differentiate services based on access to other services, such as Realvideo and SSL, from your web pages. Still others base it on how many hits your web pages get and such things. With Linux traffic control in place, a new dimension is added to differentiating services. This presents many new opportunities for differentiating services offered to your customers. For example, if you offer virtual web hosting, you could offer four different packages:

- Service Level Agreement (SLA) 1: cost $5/month—visitors to customers' virtual servers can get up to 250Kbps coming out of the server.
- SLA2: cost $7/month—250Kbps, which can overshoot to 1Mbps between the hours of midnight and 6AM.
- SLA3: cost $9/month—250Kbps, which can overshoot to 1Mbps when bandwidth is available at any time of the day.
- SLA4: cost $50/month—up to 1Mbps of high-priority, low-latency bandwidth suitable for video and audio delivery (as well as IP telephony),

with extra filters to give very low bandwidth to low priority visitors (e.g., those who get their services free).

A wide range of creative services could be offered. The time-of-day features could easily be added by using crontab-activated scripts to change configurations.

Example

The following is an example of a Linux box with two virtual servers (web, FTP, etc.) and how an ISP can sell them as two separate packages based on the maximum bandwidth one gets when visiting those two virtual servers.

Kernel Compile

I'll assume you know how to compile the kernel and add network and aliasing support. I used kernel 2.1.129 and a few other patches at the time of this writing. Linux 2.2 pre1 had just come out, but the patches had not made it in yet. By the time you read this, 2.2 will be out and everything I used will be included.

The first challenge is the clock source. In order to accurately determine bandwidth measurements, you need a very fine-grained clock. In Linux, the clock runs at a frequency of *HZ*, defined to be 100 for the ix86, i.e., 100 clock ticks per second which translates to a granularity of 10ms for each clock tick. On Alphas, *HZ* is defined as 1000, giving a granularity of 1ms. I would not suggest changing the value of *HZ* in the code. The TC clock sources are adjusted by editing the file /include/net/pkt_sched.h under the kernel tree and modifying the line which defines **PSCHED_CLOCK_SOURCE**. To start, I suggest leaving the clock source alone until you get comfortable with running other things. The default clock source, PSCHED_JIFFIES, will work fine on all architectures. Use PSCHED_CPU on high-end Pentiums and Alphas. The most precise and expensive clock source is PSCHED_GETTIMEOFDAY. Use this if you have a truly high-end Pentium II or Alpha. Do not try to use it on a 486.

Next, compile the kernel. Select **Kernel/User netlink socket** and **Netlink device emulation** to allow use of **netlink** so that tc can talk to the kernel. The second option is a backward compatibility option and may be obsolete now that 2.2 is out, so don't worry if you don't see it. Next, compile in all the queueing disciplines and classifiers. Although each can be selected as a module, I compiled them straight in. The selections are QoS or fair queueing, CBQ packet scheduler, CSZ packet scheduler, the simplest PRIO pseudoscheduler, RED queue, SFQ queue, TBF queue, QoS support, rate estimator, packet classifier API, routing-tables-based classifier, U32 classifier, special RSVP classifier and special RSVP classifier for IPv6.

Go through the normal procedure of compiling and installing the kernel.

Compiling and Setting Up TC

If you use glibc as I do (Red Hat 5.2), you will need to apply the glibc patch. A glibc patched source for **tc** is included (tc-glibc-patched.tgz). The major catch is to change the Makefile to point to where the kernel include file is. Typing **make** should then cleanly compile tc and ip for you. The ip-routing directory contains patches with names iproute2-*.glibc2.patch.gz. Get the latest one to match with the current tc. I downloaded iproute2-2.1.99-now-ss981101.glibc.patch.gz at the time of this writing.

tc Setup

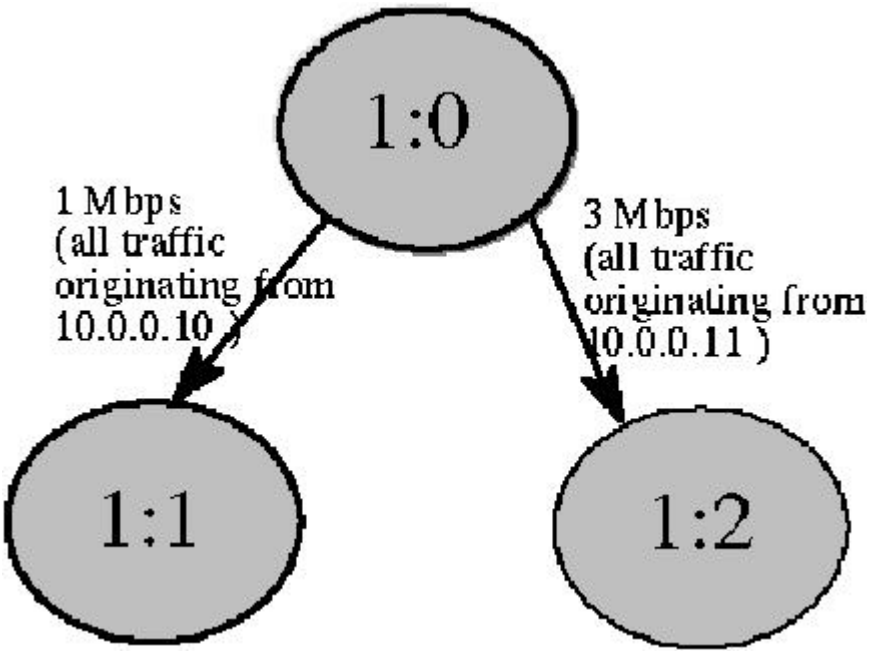


Figure 2. CBQ Tree Diagram

Figure 2 shows the simple scenario we are going to set up. Two leaf nodes are emanating from the root. IP address 10.0.0.10 (classid 1:1) and 10.0.0.11 (classid 1:2) are aliases on device eth0. They all share the same parent—classid 1:0 (the root node). Again, the intent is to show what one can do without going into fine details or building a complex TC setup. With some modifications, one can build more interesting setups with multiple devices.

The general recipe for setting up the QoS features is to first attach a **qdisc** to a device. In the sample script, this is achieved by the line

```
qdisc add dev eth0 root handle 1: ...
```

Next, define your classes. This allows you to discriminate between the different traffic types going out. In the sample script, this is achieved by the lines which start with

```
tc class add dev eth0 parent 1:0 classid X:Y ...
```

In the sample script, a one-level tree is shown. However, one can build a multiple depth tree. Basically, a child node (as shown in Figure 2) inherits from a parent and is then further resource-restricted by the class definition. For example, the root class 1:0 owns the device's bandwidth. The child node 1:1 cannot have more than 10Mbits allocated to it, but is restricted to 1Mbps. Eventually, the leaf nodes get packets sent to them based on the classifier mapping packets to them. This is quite similar to the UNIX directory and file tree structures. You can think of non-leaf nodes as directories and leaf nodes as files.

Finally, define your packet-to-class mappings to tell your classifier which packets to send to which class. You must define the classes for this to make sense. First, attach a classifier to the proper class. In the sample script, this is achieved by the construct which starts with the line

```
filter add dev eth0 parent 1:0 protocol ip ...
```

Next, define the packet-to-class mappings that will be used. In the sample script, this is defined in the constructs that define the matching criteria (such as **match ip src ...**). Always map packets to leaf classes.

If you follow this recipe and substitute the right syntax for the different queueing disciplines and filters, you will get it right. The appropriate details are in the options.

Listing 1.

Testing the Setup

In our setup, we have two virtual web servers on a single Linux machine. The setup script (Listing 1) includes some commented sample IP-aliasing examples using the supplied ip utility. The ip utility is feature-loaded and not in the scope of this article. IP addresses 10.0.0.10 and 10.0.0.11 are attached (aliased on) to device eth0 in the example.

To test, use **ftp** to get to another machine on the network. First, use ftp to get to IP address 10.0.0.10, where you should observe a rate of approximately 1Mbps. Quit that ftp session and start another one to 10.0.0.11, where you should observe a throughput of approximately 3Mbps.

Final Word

These are very exciting times for Linux. As far as I know, Linux is the most sophisticated QoS-enabled OS available today. The closest second is probably BSD's ALTQ, which lags quite a bit behind the sophistication, flexibility and extensibility found in Linux TC. I am not aware of any such functionality in Microsoft products (perhaps someone could provide pointers if they exist). Sun Solaris does have a CBQ and RSVP combo they sell as a separate product. I predict a huge increase in the use of Linux servers as a result of the many features available with TC. Alexey has taken Linux to a new level.

Support for the IETF diffserv features is also in Linux. The work extends the TC to add the most flexible diffserv support known today. Diffserv support was made possible through efforts by Werner Almesberger (who also wrote LILO, Linux-ATM and more) and myself. For more details, see http://lrcwww.epfl.ch/linux-diffserv/.

Acknowledgements

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue62/3369.tgz.

Jamal Hadi Salim (hadi@cyberus.ca) is a hacker wannabe. He spends a fair amount of his spare time staring at Linux networking code. Jamal was the original author of the Sendmail-UUCP HOWTO and is the CASIO digital diary serial driver/application author, which he still maintains. He also has sent the occasional patches to many things, including the kernel, biased towards networking issues. Currently, his efforts are focused mainly in the network scheduling code where he co-authored the Linux diffserv code with Werner Almesberger.

Archive Index Issue Table of Contents

Advanced search

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

System Administration

Martin Schulze

Issue #62, June 1999

What should you do if using a single disk or partition for the root file system is unacceptable? Use two or three. This article provides a solution for this problem.

RAID stands for “Redundant Array of Inexpensive Disks”. It is meant to speed up and secure disk access at the same time. RAID, though, is not new. It was invented in 1987 at the University of California, Berkeley. Before Linux, it was available only in the form of special hardware that was quite expensive. Of course, it could be used only in high-end computing centers.

During the last decades, performance of processing units has increased by five to ten times each year, depending on which statistic you believe. In the same period, disk capacity has doubled or tripled while prices were halved every one to two years. Used electronics don't reflect the current processor speed. This results in I/O being the current bottleneck of modern computers. Just try to compile our famous XFree86 source on a dumb PII-233 with regular SCSI disk layout.

By the time people at Berkeley realized this, they were able to foresee that no new epoch-making technology for hard disks would be forthcoming in the near future. Since magnetic- and mechanic-oriented disks were kept and laws of physics permit only slight improvements, other solutions needed to be found.

This resulted in the definition of several RAID levels. Nowadays they are used not only in high-level computer rooms, but also by the so-called middle-end sector. Since some fellow kernel hackers decided to implement RAID for the Linux kernel, this technique can be used by low-end PCs, and regular people can be satisfied by the improved performance and data security.

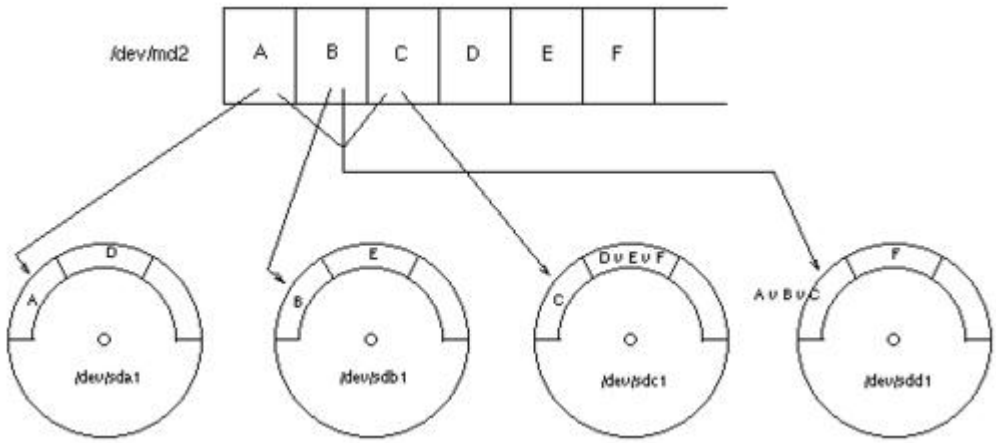


Figure 1. Operation of RAID-5

RAID levels share the following properties:

- Several different physical disks are combined and accessed as a compound element. Under Linux, this is done by the driver for multiple devices, also known as /dev/md*.
- The stored data is distributed over all disks in a well-defined way.
- The data is stored in a redundant way over the disks, so in case of failure, data is recoverable.

By dividing data into equal chunks and distributing them over all affected disks, one gets higher I/O performance than by using only one fast disk. The reason for this is due to ability to request data from the disks in a parallel fashion. The easiest way to do this is called striping mode or RAID level 0, but it doesn't contain any redundancy.

Redundancy is achieved in different ways. The simplest is to store the data on two equal disks. This is defined in RAID-1, also known as mirroring. Of course, one gets performance increase only when at least four disks are used.

More efficient redundancy is obtained when instead of duplicating all the data, a unique checksum is generated and stored with regular data. If a single disk should fail, one is able to reconstruct its data by using all data chunks of that stripe together with the calculated checksum. The easiest way to calculate a checksum is to XOR all data chunks in a stripe. This is defined in RAID levels 4 and 5. The unofficial level 6 uses another chunk for a different checksum algorithm, resulting in two redundant disks, and even better breakdown avoidance.

How to Set Up RAID

Using file systems with RAID has many advantages. First is speed. RAID combines several disks and reads/writes chunks from the disks in sequence.

Second, you can get bigger file systems than your largest disk (useful for /var/spool/news/, /pub/, etc.). Third, having achieved redundancy means a disk failure won't end up in data loss. For technical information on RAID, please refer to ftp://ftp.infodrom.north.de/pub/doc/tech/raid/.

To use RAID with Linux, you need a kernel with appropriate support. First of all, this refers to support for the “multiple devices driver” (CONFIG_BLK_MD_DEV). Linux 2.0.*x* supports linear and striping modes (the latter is also known as RAID 0). Linux kernel 2.1.63 also supports RAID levels 1, 4 and 5. If you want to use these levels for 2.0.*x*, you'll have to install the kernel patch mentioned at the end of this article.

To use either, you must activate the appropriate driver in the kernel. (I'd suggest compiling a kernel of your own anyway.) Additionally, you need to have special tools installed. For linear mode and RAID level 0, you need the **mdutils** package that should be included in your distribution. To use RAID level 1, 4 or 5, you need to have the **raidtools** package installed, which supersedes the mdutils package.

Striping works most efficiently if you use partitions of exactly the same size. Linux's RAID driver will work with different sizes, too, but is less efficient. In that case, the driver doesn't use all disks for striping after a certain amount of disk space is used. The maximum number of disks will be used at any time.

After setting up RAID and combining several disks to a compound device, you don't access the disks directly using /dev/sd*. Instead, you make use of the multiple devices driver that provides /dev/md*. These devices are block devices just like normal disks, so you simply create a file system on them and mount.

The default setup of the Linux kernel provides up to four such compound devices. Each MD can contain up to eight physical disks (block devices). If your setup requires either more combined devices or more compound devices, you have to edit include/linux/md.h within the Linux kernel source tree, especially **MAX_REAL** and **MAX_MD_DEV**. For testing purposes, you can use some loopback devices instead of physical disks.

Swapping over RAID

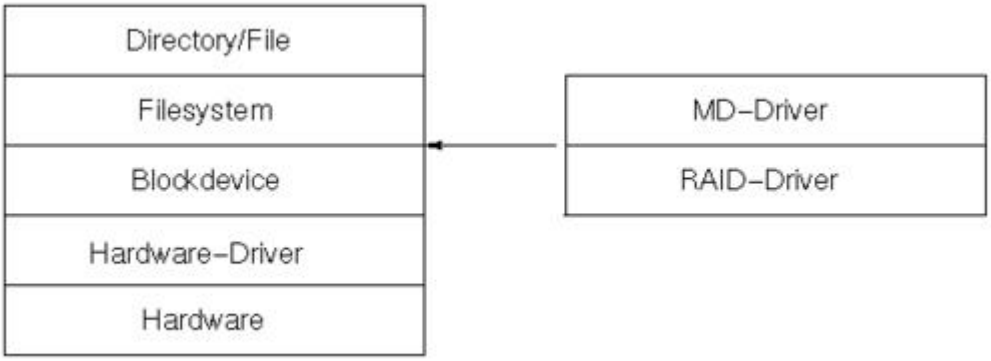


Figure 2. Position of RAID within the Linux Kernel

The Linux kernel includes native support for distributing swap space over several disks. Instead of setting up a RAID-0 device and directing swap to it, you simply add all swap partitions to /etc/fstab and use the **swapon -a** to activate them all. The kernel uses striping (RAID-0) for them. Here's a sample setup:

```
/dev/sda3  none  swap  sw
/dev/sdb3  none  swap  sw
/dev/sdc3  none  swap  sw
```

Setting Up RAID

Setting up RAID for normal file systems such as /var, /home or /usr is quite simple. After you have partitioned your disks, instruct the RAID subsystem on how to organize the partitions. This information is copied to /etc/mdtab for later activation. It can be done by issuing the following command:

```
mdcreate -c4k raid0 /dev/md0 /dev/sda1 /dev/sdb1\
   /dev/sdc1
```

Listing 1.

If you want to use RAID level 1, 4 or 5, you have to use an additional configuration file that reflects the disk setup as shown in Listing 1. These levels are more complicated and need a special signature on top of the compound device. This signature is generated by the **mkraid** command. The remaining setup looks like this:

```
mkraid /etc/raid/raid5.conf
mdcreate raid5 /dev/md1 /dev/sdd1 /dev/sde1\
   /dev/sdf1 /dev/sdg1 /dev/sdh1
```

Now two RAIDs have been created; the first consisting of three partitions, while the second uses five of them. Depending on the data to be stored on them, different chunk sizes have been selected. The next step is to activate these devices with the command:

```
mdadd -ar
```

From now on, you may refer to /dev/md0 and /dev/md1 as block devices that may contain your file systems. In order to use these devices, issue this command during the boot sequence. Please check out the startup sequence of your distribution. Some of them (e.g., Debian) have already included this line.

After the kernel knows how disks are organized, you can create your file system on the new devices and add them to /etc/fstab as usual.

Root File System on RAID

The use of RAID for the root file system is a bit tricky. The problem is LILO can't read and boot the kernel if it is not in a linear fashion on the disk (as it is on EXT2 or MSDOS file systems). The solution is to put the kernel on a different partition with a normal file system and activate RAID after the kernel is booted.

This way, LILO would boot the kernel, but the kernel itself would be unable to mount the root file system because its RAID subsystem isn't initialized yet. Now you're in trouble, right? No.

For late 2.1.*x* kernels, a kernel parameter can be used to load the kernel from a RAID.

```
md=<
   <fault level>,dev0,dev1,...,devn
```

This needs to be added to LILO using the **append=** option or directly at the LILO prompt during boot stage. You'll find more information in Documention/md.txt in the Linux source tree.

For stable kernels (2.0.*x*) and “not so late” development kernels (2.1.*x*), you need a mechanism to call some programs after the kernel is loaded but before it tries to mount the root file system. This refers at least to **mdadd**.

The only way to achieve this is to use the initial RAM disk, also known as **initrd**. General information about initrd may be found in Documentation/initrc.txt inside the kernel source tree.

You will most likely have to compile your own kernel, although you can try the one included with your distribution if it contains all facilities. You'll need to add modules support to the described solution. However, additional kernel compilation options are needed for the described setup and are shown in Listing 2. Additionally, you have to include support for your SCSI card, etc. If you're uncertain about the options, refer to the Kernel-HOWTO and use **?** to display a description of the referenced driver.

Listing 2.

If the Linux kernel uses initrd, it mounts the given RAM disk as root file system and executes /linuxrc if found. Then the kernel continues its boot process and mounts the real root file system. The old initrd root will be moved to /initrd if that directory is available or will be unmounted if not. If it is only moved, the RAM disk remains in memory. So on systems with little memory, you should get the kernel to remove it entirely when it is no longer needed.

The initrd file is a “simple” root disk, containing all the files needed for executing the /linuxrc file. If it is a shell script, it includes a working shell and all tools used in this script. In order to execute programs, it also includes a working libc with ld.so and tools. Alternatively, you can link the included programs statically and don't need a shared libc. Since this doesn't save any space, it is not necessary.

After RAID has been initialized from /linuxrc, you must tell the kernel where its new root file system resides. At that time, it may be configured to use initrd as the root file system. Fortunately, our fellow kernel hackers have designed another easy interface to set the root file system.

This facility makes use of the /proc file system. The device number of the new root file system must be sent (use **echo**) to /proc/sys/kernel/real-root-dev, and the kernel continues with that setting after /linuxrc completes.

As LILO normally isn't able to boot from a non-linear block device (such as RAID), you must reserve a small partition with the kernel and initial RAM disk on it. I've decided to use a 10MB partition as /boot. Binaries can be stored in this partition and it can be accessed from a rescue floppy. I wonder why one should use this since Linux is so stable, but for the sake of security it may be a good idea.

10MB is plenty of space for one kernel and a RAM disk of approximately 1MB in size. Currently, my system uses only 2.5MB of this space, so plenty of playing room is available. Due to the fact that /boot uses the normal file system (e.g., EXT2), you can use /etc/lilo.conf to point to /boot/vmlinuz in your setup.

Now you need to decide what to do in your /linuxrc script. Basically, activate RAID and tell the kernel where your root file system resides. Listing 3 shows an example /linuxrc program.

Listing 3.

Any block device can be used as the root file system. In the given example, 0x900 is used. This stands for major number 9 and minor number 0, which is the encoding for /dev/md0.

Next, make a list of binaries and additional files needed, including some device files in /dev. To get the /linuxrc script working, you need /dev/tty1. Other necessary devices depend on your /etc/mdtab file. You will need at least /dev/md0.

The above example uses these binaries: ash, mount, umount and mdadd. These files are also needed: mdtab, fstab, mtab and passwd.

Listing 4.

The mdtab file I use is shown in Listing 4. For my file, these block devices must be created on the initial RAM disk: /dev/hda2, /dev/hda4, /dev/hdb2, /dev/hdb4, /dev/md0, /dev/md1, /dev/md2 and /dev/md3. Use the **mknod** command to create these device files. You'll find their major and minor numbers by investigating your /dev directory or by reading Documentation/devices.txt from the kernel's source directory. The following commands create tty1 and md0:

```
mknod dev/tty1 c 4 1
mknod dev/md0 b 9 0
```

The best way to create the initial RAM disk is to create the directory /tmp/initrd and install everything you need in it. Once you have done that, determine the used disk space (**du -s**), then create the initrd file. The following commands will create an initial RAM disk 1MB in size. To use it, your kernel must include support for the loopback device.

```
dd if=/dev/zero of=/tmp/initrd.bin bs=1024k\
   count=1
mke2fs /tmp/initrd.bin
mount -o loop /mnt /tmp/initrd.bin
```

Since dynamically linked binaries are used, the Linker and the dynamic libraries must also be installed, at least /lib/libc*.so, /lib/ld-linux.so.2, /lib/ld-2.0.*.so and an appropriate /etc/ld.so.config file—appropriate means that **/lib** should be the only line in the file. Create a new library cache /etc/ld.so.cache file with the command

```
ldconfig<\!s>-r<\!s>/tmp/initrd
```

and install the needed binaries in appropriate directories: /sbin or /bin.

Don't forget to create the /proc directory, or the mount will fail. The fstab and mtab files can be empty and read-only, but must exist on the initial RAM disk. No program will attempt to write to these two files. For the /etc/passwd file, it's sufficient to include only the root user.

After you have copied everything from /tmp/initrd to the RAM disk mounted at /mnt (see above), unmount it (e.g., with the command **umount /mnt**) and move

the file to /boot/initrd.bin. Now tell LILO to load the kernel and the RAM disk, using a record in /etc/lilo.conf similar to the one shown in Listing 5.

Listing 5.

Issue the **lilo** command, and you are almost done. Since the RAID subsystem is now configured at boot stage before any /etc/init.d scripts are issued, you should disable the mdadd call in the /etc/init.d scripts.

This setup implies you have a running Linux system installed on some non-RAID disk. At least install a small base system on your swap partitions, compile the kernel on a different machine, set up RAID on the appropriate machine, move the files and continue installation afterwards.

Resources

Martin Schulze studies computer science in Oldenburg, Germany. He has used Linux for several years and tries to improve it where he can. Nowadays he maintains several machines in his hometown and is involved in several Linux projects, such as being the RAID maintainer for Debian GNU/Linux. He can be reached via e-mail at joey@infodrom.north.de.

Archive Index Issue Table of Contents

Advanced search

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

LINUX JOURNAL

The awk Utility

Louis J. Iacona

Issue #62, June 1999

An introduction to the Linux data manipulation tool called awk.

Partly tool and partly programming language, **awk** has had a reputation of being overly complex and difficult to use. This column demonstrates its usefulness without getting hung up on the complexity.

Scripting languages such as the UNIX shell and specialty tools like awk and **sed** have been a standard part of the UNIX landscape since it became commercially available. In 1982, “real programmers” used C for everything. Tools such as sed and awk were viewed as slow, large programs that “hogged” the CPU. Even applications that performed structured data processing and report-generation tasks were implemented in fast, compiled languages like C.

Part of my motivation for writing this article comes from observing that, even today, most system administrators and developers are either uninformed about or intimidated by utilities like awk and sed. As a result, tasks that should be automated continue to be performed manually (or not at all), or duller tools are used instead.

Admittedly, both awk and sed are rather peculiar tools/languages. Both recognize traditional UNIX “regular expressions”—powerful, but not trivial to learn. Both tools seem to offer too many features—quite often providing several ways of performing the same task. Therefore, mastering all the features of awk and sed and confidently applying them can take awhile—or so it may seem. First impressions notwithstanding, you can quickly and effectively apply these tools once you understand their general usefulness and become familiar with a subset of their most useful features. My intent is to provide you with enough information and example code for getting jump-started with awk. You can read about sed in April's “Take Command: Good Ol' sed” by Hans de Vreught.

sed and awk are two of the most productive tools I have ever used. I rely on them quite heavily to implement a wide range of tasks, the implementation of which would take considerably longer using other tools/languages.

I will assume you have heard of or worked with some of the more significant sub-systems of Linux and that you have an understanding of how to use the basic features of the shell command line, such as file I/O and piping. Familiarity with a standard editor such as **vi** and a working knowledge of regular expressions would also be useful. Many Linux commands, including **grep**, awk and sed, accept regular expressions as part of their invocation, so you should at least learn the basics.

A Word about Regular Expressions

My coverage of the awk tool is limited to an introductory foundation. Many advanced features are offered by awk (gawk and nawk) but will not be covered here.

General Overview

The meaning behind the name of this tool is not terribly interesting, but I'll include an explanation to solve the mystery of its rather uncommon name. **awk** was named after its original developers: Aho, Weinberger and Kernighan. **awk** scripts are readily portable across all flavors of UNIX/Linux.

awk is typically engaged to reprocess structured textual data. It can easily be used as part of a command-line filter sequence, since by default, it expects its input from the standard input stream (stdin) and writes its output to the standard output stream (stdout). In some of the most effective applications, awk is used in concert with sed—complementing each other's strengths.

The following shell command scans the contents of a file called **oldfile**, changing all occurrences of the word “UNIX” to “Linux” and writing the resulting text to a file called **newfile**.

```
$ awk '{gsub(/UNIX/, "Linux"); print}' oldfile \>\
newfile
```

Obviously, awk does not change the contents of the original file. That is, it behaves as a stream editor should—passively writing new content to an output stream. This example barely demonstrates anything useful, but it does show that simple tasks can be implemented simply. Although awk is commonly invoked from a parent shell script covering a grander scope, it can be (and often is) used directly from the command line to perform a single straightforward task as just shown.

Although awk has been employed to perform a variety of tasks, it is most suitable for parsing and manipulating textual data and generating formatted reports. A typical (and tangible) example application for awk is one where a lengthy system log file needs to be examined and summarized into a formatted report. Consider the log files generated by the **sendmail** daemon or the **uucp** program. These files are typically lengthy, boring and generally hard on a system administrator's eyes. An awk script can be employed to parse each entry, produce a set of category counts and flag those entries which represent suspicious activity.

The most significant characteristics of awk are:

- It views its input as a set of records and fields.
- It offers programming constructs that are similar (but not identical) to the C language.
- It offers built-in functions and variables.
- Its variables are typeless.
- It performs pattern matching through regular expressions.

awk scripts can be very expressive and are often several pages in length. The awk language offers the typical programming constructs expected in any high-level programming language. It has been described as an interpreted version of the C language, but although there are similarities, awk differs from C both semantically and syntactically. A host of default behaviors, loose data typing, and built-in functions and variables make awk preferable to C for quick-prototyping tasks.

awk Invocation

At least two distinct methods can be used to invoke awk. The first includes the awk script in-line within the command line. The second allows the programmer to save the awk script to a file and refer to it on the command line.

Examine the two invocation styles below, formatted in the typical man page notation.

```
awk '{
awk -Fc -f script_file [data-file-list ...]
```

Notice that *data-file-list* is always optional, since by default awk reads from standard input. I almost always use the second invocation method, since most of my awk scripts are more than 10 lines. As a general rule, it is a good idea to maintain your awk script in a separate file if it is of any significant size. This is a more organized way to maintain source code and allows for separate revision control and readable comment statements. The **-F** option controls the input

field-delimiter character, which I will cover in detail later. The following are all valid examples of invoking awk at a shell prompt:

```
$ ls -l | awk -f
$ awk -f
$ awk -F: '{ print $2 }'
$ awk {'print'} input_file
```

As you will see through examples, awk programming is a process of overriding levels of default actions. The last example above is perhaps the simplest example of invoking awk; it prints each line in the given input file to standard output.

The Language

If you acquire a thorough understanding of awk's behavior, the complexity of the language syntax won't appear to be so great. To provide a smooth introduction, I will avoid examples that take advantage of regular expressions (see “A Word About Regular Expressions”). **awk** offers a very well-defined and useful process model. The programmer is able to define groups of actions to occur in sequence before any data processing is performed, while each input record is processed, and after all input data has been processed.

With these groups in mind, the basic syntactical format of any awk script is as follows:

```
BEGIN {

}
{

}
END {

}
```

The code within the **BEGIN** section is executed by awk before it examines any of its input data. This section can be used to initialize user-defined variables or change the value of a built-in variable. If your script is generating a formatted report, you might want to print out a heading in this section. The code within the **END** section is executed by awk after all of its input data has been processed. This section would obviously be suitable for printing report trailers or summaries calculated on the input data. Both the **END** and **BEGIN** sections are optional in an awk script. The middle section is the implicit main input loop of an awk script. This section must contain at least one explicit action. That action can be as simple as an unconditional print statement. The code in this section is executed each time a record is encountered in the input data set. By default, a record delimiter is a line-feed character. So by default, a record is a single line of text. The programmer can redefine the default value of the record delimiter.

The following input data text will be assumed in each of the following examples. The content of the data is somewhat silly, but serves the exercise well. You can imagine it representing a produce inventory; each line defines a produce category, a particular item and an item count.

```
fruit: oranges 10
fruit: peaches 11
fruit: plums 11
vegetable: cucumbers 8
vegetable: carrots
fruit: tomatoes 2
```

We will start off very simply and quickly work into something non-trivial. Notice that I make a habit of always defining each of the three sections, even if the optional sections are stubbed out. This serves as a good visual placeholder and reminds the programmer of the entire process model even if certain sections are not currently useful. Be aware that each of the examples could be collapsed into shorter scripts without any loss of functionality. My intent here is to demonstrate as many awk features as possible through these few examples.

Listing 1.

Look at the example script in Listing 1 and try to relate it to its output:

```
fruit: oranges 10
fruit: peaches 11
fruit: plums 11
fruit: tomatoes 2
```

By default, an input record is a line-feed terminated section of text, so if the input contains six lines, the implicit main loop marked by the **# (1)** comment executes six times. The awk source-code comments are specified with a # character—the interpreter ignores characters from the # to the end of the line (same comment style as the UNIX shell). The built-in variable **$0** always contains the entire current record value (see built-in variable table below). The line below the (1) marker checks to see if the current input record is an empty line. If it is, awk goes on to read the next input record. Each field within a record is assigned to an ordered variable—**$1** through ***$N*** where *N* is equal to the number of fields in the current record. What determines a field? Well, the default field separator is any "white space"—a space or tab character. The field separator character can be redefined. The line below the **# (2)** comment will print out the entire record if the first field is set to **fruit:**. So, when looking at the output produced by Script 1, all lines of type **fruit** are displayed.

Listing 2.

Take a look at the example script in Listing 2 and try to relate it to its output below. The only noticeable enhancement is the data summary at the end—stating how may of the total units were of type **fruit**.

```
fruit: oranges 10
fruit: peaches 11
fruit: plums 11
fruit: tomatoes 2
4 out of 5 entries were of type fruit:.
```

This time, we made use of the two optional **BEGIN** and **END** sections of the awk script. The group of statements preceded by the **# (1)** comment initialize some programmer-defined variables: **FCOUNT**, **COUNT** and **TYPE**—representing the number of **fruit:** records encountered, the total number of records and the produce-category name. Notice that the line preceded by the **# (3)** unconditionally increments the record counter (also note that syntax is borrowed from the C language). The section of code preceded by the **# (4)** comment now references the **TYPE** variable instead of a literal string, and increments the **FCOUNT** variable. The next section of code makes use of the **printf** built-in function (works just as the C-library printf does, but differs a bit syntactically) to print out a sub-count and a total count.

Listing 3.

Look at the example script in Listing 3 and try to relate it to its output. Notice that the only records displayed are those which were flagged as an error and those indicating a supply shortage. The summarization at the end of the output now includes additional information. Output from Listing 3:

```
Parsing inventory file "input_data"
Bad data encountered: vegetable: carrots
Short on tomatoes: 2 left
4 out of 5 entries were of type Fruit.
1 out of 5 entries were of type Vegetable.
0 out of 5 entries were of type Other.
1 out of 5 entries were flagged as bad data.
1 out of 5 entries were flagged in short supply
```

In this third example, we make further use of the two optional **BEGIN** and **END** sections. Once again, the **BEGIN** section initializes some programmer-defined variables. It also prints out a heading that indicates the name of the input file (the built-in variable **FILENAME** is referenced). Notice the new code section preceded by the **# (3)** comment. The **NF** variable is a built-in that always contains the number of fields contained in the current record. Since white space is still our field delimiter, we would always expect three fields. This code section flags and displays a record that is deemed bad data. Also, a counter maintaining the number of errors is incremented. Since records deemed invalid are useless, the program then goes on to process the next input record. The code section preceded by the **# (5)** comment was altered to maintain additional counts based on the produce category type.

Now let's assume a system administrator is asked to determine the proportions certain shell interpreters are being used with the choices of the standard Bourne Shell, the Korn Shell and the C Shell. The script will provide a

breakdown of usage by total count and percentages and flag the instances where a login shell was not applicable or not assigned to a system user. Examine the script in Listing 4—it satisfies our requirement. Relate the code to its output in Listing 5.

Listing 4.

Listing 5.

The first thing worth noticing in the Listing 4 script is the assignment to the built-in variable **FS**—the input field delimiter. Entries in the /etc/passwd file are made up of colon separated fields. Field 7 indicates which program (shell) is run on behalf of that user at login time. Entries with an empty field 7 are printed out, then the summary report is printed.

Thus far, we have reviewed awk's behavior through several small examples of code. The features demonstrated provide a working foundation. You have seen the execution flow of an awk process. You have seen built-in and user-defined variables being manipulated. And you have seen a few built-in awk functions applied. As with any high-level language, one can be very creative with awk. Once you get comfortable, you will want to put it to more sophisticated use. Most Linux systems today offer the features of **nawk** (new awk), which was developed in the late 1980s. nawk and GNU's **gawk** make it possible to do the following within an awk script:

- Include programmer-defined functions.
- Execute external programs and process the results.
- Manipulate command line arguments more easily.
- Manage multiple I/O streams.

Table 1.

Table 2.

As a reference, Tables 1 and 2 define the most common built-in variables and functions. Also, note that the following operators each have the same meaning in awk as they do in C (refer to the awk man page):

*** / % + - = ++ -- += -= *= /= %=**

Conclusions

Scripting languages and specialty tools that allow rapid development have been widely accepted for quite some time. Both awk and sed deserve a spot on any Linux developer's and administrator's workbench. Both tools are a standard

part of any Linux platform. Together, awk and sed can be used to implement virtually any text filtering application—such as perform repetitive edits to data streams and files and generate formatted reports.

The most current reference book for both awk and sed is the O'Reilly release *awk and sed* by Dale Dougherty and Arnold Robbins. Also see *Effective AWK Programming* by Arnold Robbins (SSC). For an immediate on-line synopsis on your Linux system, use the **man** command as follows:

I hope the information provided here is useful and encourages you to begin or expand your use of these tools. If you exploit what awk and sed offer, you will most certainly save development time and money. Those who know how to quickly apply sharp tools to seemingly complex problems are handsomely rewarded in our field.

Louis Iacona (lji@peakaccess.com) has been designing and developing software systems since 1982 on UNIX/Linux and other platforms. Most recently, his efforts have focused on applying World Wide Web technologies to client/server development projects. He currently manages the Internet development group at ICP in Staten Island, N.Y.

Archive Index Issue Table of Contents

Advanced search

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

Letters to the Editor

Various

Issue #62, June 1999

Readers sound off.

I18N

I work as a journalist for the Slovak magazine PCRevue, so typing articles in my native language is very important for me. On page four (describing the cover) of the 3/99 issue of *LJ*, you stated that Linux is truly an international OS. A true statement, but just partially.

Stephan Kulow, Mathias Elter and I along with many translators from around the world want to make the KDE environment internationalized. It works fine for us; we have menu items, and I can write ISO-8859-2 characters to this localized Kmail window with no problems. KDE was the first desktop environment in Slovakia, translated to our language (Windows 95 was not translated; they just did Windows 98). This is one area in which we are a step ahead. GNOME, CDE and other desktop systems for UNIX have not been translated.

Also, the developers of (mainly) commercial software don't handle I18N properly, so we must use wrappers for libX11 done by Stano Meduna, et al. We have some patches that would help I18N in XFree, for example, but XFree development isn't truly open—no reply, no help. No truly usable office software is available under Linux for our language. For example, Star Office from Star Division doesn't initialize I18N properly. With wrappers, it works partially, but when importing MS Word 97 documents, it won't change the encoding from Windows-1250 to ISO-8859-2, so I am displaying Windows-1250 with ISO-8859-2 fonts. Export of HTML is done via that ugly 7-bit convention. Surprisingly, not all characters are working like this (e.g., l-acute works only in some browsers and doesn't get exported correctly). WordPerfect took care in providing I18N support, but it also uses the 7-bit convention for HTML export and cannot import other HTML encodings (like Unicode). The import/export of Word works

fine, but it is not perfect. The t-acute and l-acute characters are changed for question marks; l-acute is probably used only in our language, but programmers should think about it.

I am looking for Koffice or some open-source office package which I could fix to use I18N correctly, or a commercial company that fixes its errors fast, responds to users and takes care about their products. We have found, for example, that Red Hat is such a company. They have incorporated Stano's patches for X and our key maps into the distribution along with character sets and other things to help I18N. But they don't make an office suite.

—Juraj Bednar bednar@isternet.sk

Table of Contents for Supplements

Two possible solutions to the weird rules of the Post Office are: print the TOC in the previous or following month's magazine, or publish the TOC on your web site.

—Alan Rocker rocker@acs.ryerson.ca

As a matter of fact, we do publish the TOC on-line. See http://www.linuxjournal.com/issue57/lstoc.html for both the TOC and all the articles
—Editor

Interactive *LJ*

Ah, you just robbed me of all the fun of making my own searchable index of *LJ* issues. Your service at http://interactive.linuxjournal.com/ just found the article I needed.

Thank you for a very useful on-line service, and of course, the indispensable *Linux Journal*.

—Rolf Magnus Nilsen rolf@itdata.no

Error in March 1999 issue

I always take the “Best of Technical Support” column with a grain of salt, since it is user opinions based on no investigation of the specific situation. However, one in March was kind of appalling. Luis Cardenas asked if PCI modems would work with Linux. You published a response by Mario Bittencourt that was a very nice, concise guide to making ISA modems work with 2.0.*x* or older kernels.

Unfortunately, the advice is not at all applicable to PCI modems and could very well lead people to go out and purchase such modems, only to find they won't work on Linux.

Further, the advice is only good for older kernels which still support the deprecated /dev/cua# devices. It has been considered bad practice to use these devices for some time, and in the current kernel they don't exist anymore.

—Shawn McMahon smcmahon@eiv.com

Yes, you are correct—PCI modems are basically Windows only. We should have caught that, sorry —Editor

Kudos to Trident

I am writing to let you know of a recent hardware company's exceptional support to the GNU/Linux community and the GPL.

The Advanced Linux Sound Architecture project (http://alsa.jcu.cz/) is a project designed to build an architecture for pro-quality sound and MIDI applications, from low-level drivers for sound and MIDI hardware to high-level libraries and sequencers. The project is committed to releasing all work under the GPL.

As you may know, many sound card manufacturers are reluctant to give any technical help, and even some of those that offer help require non-disclosure agreements, which exclude the possibility of released source. We have blacklisted some companies (http://alsa.jcu.cz/black.html) which have either refused to release information or have decided to release binary-only drivers, which ALSA will not use.

Trident (http://www.tridentmicro.com/) recently contacted the ALSA developer mailing list, having written their own drivers for their 4DWave chip set for ALSA, and offering the source for the drivers. They graciously allowed all of it to be put under the GPL, including technical documents.

I am hoping to drum up support for their hardware in order for the community to demonstrate how cooperation of this sort can aid sales. Maybe this will convince more companies to follow.

Their chip set is used in the following products. If GNU/Linux users are looking towards purchasing a sound card, perhaps they would consider some of the following, since these cards are well-supported under ALSA.

Company	Product Name
=========	============================
Best	Union Miss Melody 4DWave PCI
HIS	4DWave PCI
Warpspeed	ONSpeed 4DWave PCI
AzTech	PCI 64-Q3D
Addonics	SoundVision (model SV 750)
CHIC	True Sound 4Dwave
Shark	Predator4D-PCI
Jaton	SonicWave 4D
Paradise	WaveAudio Interactive (Model AWT4DX)
Promedia	Opera CyberPCI-64
Stark	PCI

You can read more about ALSA and the call to sound card manufacturers at http://alsa.jcu.cz/call.html/.

—Thomas Hudson thudson@cygnus.com

Window Manager Confusion Spells Doom

My company sells CAD software for Windows NT, we run the company on Macintoshes, and we serve our data from Linux. I was surprised how easy Linux was to get up and running.

However, once upon a time there were two UNIX tribes: the Open Software Foundation and UNIX International. Each tribe was trying to rubber-stamp UNIX for the masses. As the two camps fought, a third tribe from the Northwest called Microsoft provided business with Windows NT. Windows NT did not provide anything great; in fact, it was largely based on an operating system

from Digital Equipment Corporation called VMS. The UNIX tribes had beaten this operating system years before.

Linux has a real opportunity to compete in business beyond the server. All that is needed is a single standard desktop. Forget about KDE and GNOME. Merge them, do whatever you must with them, but get a single desktop environment for which developers can write applications and users will prosper.

I am convinced Windows NT cannot serve my business. Unfortunately, until Linux provides a common desktop, Apple's OS X may be the only way to finally bring the power of UNIX to the business user.

—Jeff Millard President, SolidVision, Inc. jeffmill@solidviz.com

Linux Certification

I am writing in regards to the article “Linux Certification for the Software Professional” by P. Tobin McGinnis in *LJ* April 1999. First, I have been working with PCs since I got my first Apple back in the early 80s. When I went to college, I started using the x86 platform. After finishing my college degree, the Air Force sent me to Keesler AFB to teach computers to new recruits. There I first heard the word “certification” in regards to Novell's CNE program. We were teaching Novell 4.*x* to our students. People started talking about Novell certification, and of how much one could make if they had this certification. A couple of years later, I was working as a contractor for the Corp of Engineers in Mobile, AL on a team that went to field sites to convert Novell servers to NT. Once again, I heard the certification buzzword, this time from Microsoft, and the buzz of big money if one had the MCSE certification.

People began to scramble to get certified at great cost to themselves—$100 US per test, $50 US per book, and in some cases, thousands of dollars for classes. Now, many of the questions and answers for these certification tests can be found on the Web. People are passing the test who have never installed the NT OS. Yet because they are certified, they are called experts—their certificate is meaningless.

Now I read in *Linux Journal* the buzz about Linux certification. I thought Linux was better than that. Certification has always been a way for companies to market their products. The training companies are the only ones getting rich from certification. Linux has grown exponentially without certification. I am against certification.

Please don't promote Linux certification. It will ruin the Linux movement. The movement was created by people who rebelled against the likes of MS, Novell

and many other companies. Now it seems we are running to join their ranks with the same certification nonsense.

—Kyle Enlow kenlow@indiana.edu

Linux in Schools

I am writing this after reading the article “Linux in a Public High School” by Andrew Feinberg in your March issue. I am currently one of two system administrators at Bridges Academy, located in Los Angeles, California. My associate, Brook Elliot, and I run a Debian 2.0 box with the 2.0.36 kernel. The Linux box runs on a T1 line, and using DHCP, we enable students to plug in to any of the Ethernet jacks around the school and check their e-mail or browse the Web. Although most of these students are using Windows 9*x* machines, a select few of us use the full Linux potential.

Both Brook and I are students at this school. We currently have a web page up for the school at http://www.bridges.edu/, which runs under Apache.

—Matthew Kaufman xel@Bridges.Edu

Byte Magazine Syndrome

Many years ago, *Byte* magazine was a technically oriented magazine that I enjoyed reading as I enjoyed reading the December issue of *Linux Journal*. Early last year, *Byte* was publishing fluff. I hope the March 1999 issue of *Linux Journal* is not an indication of more fluff to come. With the exception of the GNU **gettext** article, where are the items that show a reader *how* to use a Linux system in a new or more productive way, to use a new programming library, to reduce incoming spam, etc.? One or two articles on interesting uses of Linux is fine (such as “Linux for the International Space Station Program” by Guillermo Ortega), but please don't go the way of *Byte*.

—Richard Film. Unix. Mirkwood@disney.com

I wouldn't call discussions of Internationalization “fluff” myself, but everyone has his own opinion. As always, we try to remain as balanced as possible between technical and non-technical articles. I hope April and May were more to your liking —Editor

Archive Index Issue Table of Contents

Advanced search

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

More Letters to the Editor

These letters were not printed in the magazine and appear here unedited.

Network Administration with AWK

Having written many administration tools in AWK over the years, I was pleased to see an article in your latest issue with that title. Too bad that's not what the article was about. Further a shame that every code snippet he included merely pulled in some external data and sent it to stdout. And to further suggest that one use grep to extract the data! AWK deserves so much more.

—Chip Christian, chip@princetontele.com

Feature Page

With the help of the March issue of the *Linux Journal* I have been able to write a feature page for the Sheffield Linux User Group site.

The authors of the original authors have written to me to help me to complete the Sheflug feature for April. I will upload the pages at the end of the month. I would like to say thank you very much to the staff at *LJ*. If I hadn't subscribed to the *LJ* I wouldn't have been able to produce such an interesting article for Sheflug.

The proposed subject for the May feature is VNC servers with a quick mention of VM Ware.

—Richard Ibbotson, r.ibbotson@zen.co.uk

Puffins and H-P

The Thursday *Wall Street Journal* article, “The Puffins Guide H-P Into an Alien Culture On a Software Mission”, reminds me of an elephant dancing with a mouse. I wonder which one is larger; the multi-million dollar corporation or the Linux internet culture. Time will tell.

Christopher Beard and Alex deVries were wise not to ask for any money. Just ask Ron Jones owner of Colossal Graphics and developer of a bulk ink system who sued H-P for reneging on a deal. He was awarded $6.4 million-dollars by the Hon. Joseph F. Biafore on January 15, 1999.

Sorry fellows, there is no check in the mail!

—Gerald Green, GG10MAN@aol.com

Error in April issue in "grep:searching for words" article

In this article Jan states “grep also accepts regular expressions..... the following command can be given grep flip *”.

Of course, grep does indeed process regular expressions, but the example above is not a demonstration of this ability.

It is misleading because in this example grep does not see any regular expressions. The “*” is expanded by the shell, not by grep. The shell then passes the list of filenames to the grep command.

This is an important point because newcomers to UNIX need to understand the power of the shell. Some other operating system, such as IBM OS/400, do not have facilities such as globbing built into the shell, but instead pass the responsibility to each command program.

An example of a regular expression in grep is: (a) `$ grep [pw] filename`
which lists all lines in the the file “filename” which contain either 'p' or 'w'.

Or you could have the same example, with shell globbing: (b) `$ grep [pw] file*` Which would do the same thing, but for all files starting with “file”. Say the directory holds three files named “file1”, “file2”, and “file3”, and you type in example (b) : the shell expands the command string into `grep [pw] file1 file2 file3`, forks a new process and executes grep. So grep doesn't even “know” about the use of “file*”.

Of course, shell globbing can be used with many different commands, not just grep. But beware, not all commands can process multiple files.

—Tim Lewis, timlewis_atlanta@yahoo.com

ARKEIA (fwd)

I read your article about ARKEIA and downloaded it. Since I don't use tape drives, but hard disk for my back up, I have some problems that should be noted. I set up everything by instructions, but it didn't work. When backup starts it says: insert tape /mnt/backup (this is where my backup disk is mounted). I say OK, but it's not satisfied. When I check all of the setup files I noticed, that ARKEIA for tape type “file” automatically asiges “manually mountable” flag. It can not be changed to “on-line”, since it is set automatically. So I tried with /dev/hdb1 as tape location, but without success. I think that a lot of readers of *Linux Journal* use disk's as their back-up devices. I read all the documentation (more than 300 pages), but found only one or two paragraphs dedicated to file backups. I would really appreciate it if you could try it and report it to the *LJ* for readers to know about it. I found information in FAQ, but then I noticed that I would need non shareware version to get it working (you need support for lib management, which is not included). I contacted ARKEIA

support and they confirmed it. I told them that I find it strange limitation for the shareware version and they agreed to think about it. Support was, as you said, very quick and friendly. But facts are facts and I thing *LJ* readers should know about it.

Have fun!

—Bostjan Vlaovic, bostjan.vlaovic@uni-mb.si

Re: ARKEIA (fwd)
Hi, Bostjan.

I don't have the docs for Arkeia handy, so I'll comment on this from memory. Arkeia lets you define files of various sizes as “tapes”, which you then define as elements of a drivepack. I haven't tried this recently but I believe the version I used when I wrote the review would let me back up to files. If so, can you define a directory structure on your backup hard drive that lets you rotate the backups?

I'll experiment with this next time I get a chance & let you know what I find. If you get that working, please let me know.

Then there's the old standby, tar and gzip. Most backup programs compress each file individually so that they can then recover any one file later at restore time. However, if you tar then gzip, gzip compress the entire tar file as one file. This results in better compression, but may require a temporary file later at restoration time. If you have the hard drive space to do it, you might find that will do what you want.

—Charles Curley, ccurley@trib.com

Oh no . Not Another Microsoft.
Dear Linux journal Editor and Netscape team.

I would like to comment on the Letter Red Hat Phenomenon which was published in the March 199 issue. I think that Mr. Reilly is right and that you are not taking this matter seriously enough. Do we want another Microsoft in face of RedHat Linux Monopoly.

At Netscape's website I saw the advertisement of their LDAP server . http://home.netscape.com/download/noframes_102_21.html#specs Netscape directory Server was advertised to be workable on Linux.

After downloading it and going to the installation page I found out to my surprize that it works with RedHat Linux only. http://home.netscape.com/eng/server/directory/4.1/installation.html

I tried to install it on SuSE Linux 5.3 and 6.0 and nope, didnt work.

This is scary.

I dread of the day when products would say .

```
Installation Requirements:
Solaris 2.6
Windows NT 5
Red Hat Linux x.x
```

PS:: RedHat Linux is just of the distributions but Linux is not just RedHat Linux. I think that this matter is very serious. I really like Linux and its openness. I love to see different packagers packaging Linux to their own style: redhat, suse, slack, debian... etc but I would hate Linux to divide and not be even portable among Linuxes. Please dont let that happen.
I think that Linux journal is great and is one of the very few journals that I really take time to read.

—Atif ghaffar, webmaster@artemedia.ch

---

letter to the editor

I would like to thank everyone who responded to my article, “Linux in a Public High School.” I would like everyone to check out my latest endeavor, the High School Linux User Group. The HS-LUG right now is just a few web pages and a mailing list, but I would like people to have somewhere to turn when they want to ask people about the uses of Free Software in education. The HS-LUG is at http://hs-lug.tux.org.

—Andrew G. Feinberg, andrew@ultraviolet.org

Linux Journal

A few notes:

1) Ethernet frames range from 64 to 1518 bytes (1514 misses the CRC).

2) CSMA/CD's “capture effect” is largely a figment—in many years consulting on thousands of networks I've yet to see it for more than some milliseconds, primarily because systems do not have resources to send at continuously full rate and a collision randomizes things quickly—yes, in some testbeds, you can demonstrate one sender's dominance for a while, but you have to work at getting the conditions right. Years ago there were chip sets (e.g., from Seeq) that violated the inter-packet gap spec, so indeed could capture a net—Cisco loved them—but the uproar was so great that Seeq decided to back off (no pun).

3) Dropping frames indeed slows down TCP a lot, due to its inability to distinguish error loss from router-congestion loss. However, this removes traffic, rather than “...leads to more congestion”. The timeout and retransmission times start at about a second and double up to about 32 seconds, whereupon transmission stops and higher layers are informed of a network error. The attached Powerpoint graph of a real TCP situation, where only errors caused loss, shows how significant delays (lower traffic) occur after each loss—blank spaces between triangles and

upward drifts in bytes that could have been sent but weren't. As a protocol, TCP still needs help in these respects.

4) “...the physical distance between the furthest nodes...”—“farthest”. The editor should know that “further” refers to process (she furthered his election) and “farther” refers to physical distance.

Again, good article.

—Dr. A. B. Cannara, cannara@ibm.net

In Defense of Red Hat and other subjects!

First let me introduce myself. I am H. Aurag, a Ph.D. student at Univ of Montréal (math). There, I have been using for a while different Unix machines, from dummy terminals to SGI and Sun Workstations.

On my HOME computer, I have Windows95/RedHat Linux installed since 1998. Needless to say what I think of Windows95. The only reason I am keeping it is StarCraft, and some home software.

I think I might be the only real home user of Linux, in that I never use it elsewhere (I have no choice or say there). To tell the truth, going from Unix (Iris, Solaris) to Linux was not that hard for me as opposed to someone who would go from a Win95 only environment. But the first thing I noticed was that Linux is NOT Unix. It looks like unix, uses shells and X etc... BUT it's not Unix. And the surprise was that I found it superior, for the use I make of it (Math computations symbolic and numerical plus TeX, web surfing, and soon Civilisation CTP), than Iris or Solaris. I even found that my lousy 166MMX machine was faster than some Ultra Sparc workstations: I actually did some benchmarks, and had a big laugh.

This is the first thing I wanted to say to Unix users. You will feel comfortable with Linux, but PLEASE don't think of it as a cheap Unix clone. IT IS NOT. And I hope it will never be. Some distros might want to let you think that, but I don't agree. My hope is that Linux evolves as a unique and powerful OS (it is already) like it did by incorporating the best ideas from anywhere even NT if need be!

Also, I think Red Hat has made a great job (I really don't know any other distro, and I am not interested in knowing them). They gave me something I could use and install easily, something that makes my 166MMX with 64Meg machine faster than an Ultra5 with 96Meg and 270Mhz RISC and ALL that uses Solaris. When I am saying faster, I am talking about programs that make heavy symbolic computations and heavy floating point operations!

So kudos for RedHat, for KDE, for GNOME (I like it most! Excellent job!), for all those programmers that make nice programs and nice GUI's for

newbies, also to Debian, Slackware, Suse... even if I don't know them. They all work hard I think!

Let's hope Linux keeps up with the good work and I hope I will be able to have all those games browser plugins and I hope StarOffice will realease a 5.1 version that uses less memory ;)

Finally, I would like to say to all those newbies out there, that a full RedHat install takes 40 minutes or less, and a full Win95/98 install takes 2 hours! (If you read the manuals that came with both OS'es before installing).

—H. Aurag, aurag@geocities.com

---

grep -c

In *LJ* April 1999 Take Command article on the grep command the writer states that the -c option “outputs the number of times a word exists in a file”. According to the man page the -c option “print a count of matching lines”. This difference will be apparent if the word appears more than once on the same line.

Thanks for a great magazine.

—David Massey, david.massey@ub.uio.no

Linux in Brunei

Just today, a member of the Brunei-Linux Users Group emailed me, informing another fellow Bruneian had sent off an email to the *Linux Journal*.
It appears that dear Stefanus Du Toit [sdt@ultracool.net] is loneley, thinking he is one of the few people who use Linux in Brunei.

Well, dear Stefanus !! Be lonely no more !!

Brunei-LUG needs people such as yourself who are willing to collaborate with other users, and to advocate the use of Linux within Brunei.

We also deal in the distribution of GPL'd Linux Distributions, please contact us for the latest distributions: Red Hat 5.9, 5.2, Debian 2.1, Slackware 3.6, Slackware 4.0 beta etc etc.

Check us out at: http://irb.lh.umist.ac.uk

oh, and if you check up the Linux counter, you might notice something interesting:
http://counter.li.org/bycountry/BN.html

Shows it's just the two of us :)

—Izam-Ryan Bahrin, izamryan@netscape.net

Minor correction and Letter to the Editor

Linux Journal is amazing, and issue #60 was no exception. However, I'd like to point out a possible minor typing mistake on article “Linux 2.2 and the Frame-Buffer Console”, page 14: under the “Frame Buffers” subtitle, on the 2nd paragraph, the text says the frame buffer device files are “/dev/fd*”. Would them be “/dev/fb*”?

On the same *LJ* issue, reading the Letters To The Editor section, the letter “SSC's Distribution Choice” raised an interesting point for me: the fear of having the Linux software market being directed to a specific Linux distribution. Although we know that Linux is just the OS kernel, users frequently see Linux as an entire package, from the installation program to the applications that come with a distribution. So, there are of course chances that a specific distribution be “preferred” by the general Linux user. In my opinion, when a company says it will support a specific commercial distribution, either that distribution has a special characteristic (such as glibc), or there is contribution from the distribution's company side on porting their products to Linux. I am not a RedHat or Caldera fan, but I must admit these are examples of companies that worked together with some commercial software developers to port their products to Linux. Anyway, I believe that any Linux software geared towards a specific distribution can be run on any other distribution. What is needed is some standardization between distributions, such as to include or not to include glibc, to use or not to use .RPM packages, the filesystem layout, and so on.

Sorry for a so long letter.

—Celso Kopp Webber, webber@sj.univali.rct-sc.br

Certification

I totally disagree with P. Tobin Maginnis, article “Linux Certification for the Software Professional”, April 99 *Linux Journal*. I am actually a little disappointed in you for publishing it. I feel that some readers who don't think too deeply about what is presented will be seduced into agreeing.

In the opening paragraph Maginnis says “ [certification] will be essential in the future as Linux software is brought into corporate and government environments.” This has not been true for any of Linux's competing o perating systems. I believe that Linux's chances of penetrating these markets will depend upon its performance and stability compared to its competition.

Instead of giving us convincing logical arguments why we need to license ourselves, he points to so many other fields that require a license as ifthis is enough reason. Next, “the U.S. Department of Labor predicts ..... 300,000 more ..... positions than programmers ..... and universities will not be able to fill the demand .....” Here he's trying to scare you and at

the same time he's hoping you won't realize that such certification will limit or reduce the number of programmers that qualify which only aggravates the problem.

“..... a fundamental concern of corporate and government managers is the cost of operations—and Linux dramatically reduces this cost.” And if we will just accept certification then one of the benefits is “ ..... more job options and ..... better pay”, thus higher operating costs.

“The consensus among managers is that certification leads to improved employee service .....”. What group of managers of “certified” programmers produced this “consensus”? Assertions like this with no supporting evidence should be regarded as false.

“Licensing is a governmental action that seeks to safeguard the health, safety, welfare .....”. I say BULL! When was the last time you saw anyone go after a bad doctor, lawyer, engineer or accountant? When was the last time you saw one doctor, lawyer, engineer or accountant testify against one of his fellows? I am sorry, but, once licensing gets in place all the license agency cares about is its own bureaucratic rules, forms and fees.

“..... the expected rapid growth of Linux will create a need for business and government managers to have an additional metric in the selection of new employees .....”. Why is a new metric required? STANDARDIZING is not an answer! If you inspect every programmer ad in a major city newspaper you will NOT find two alike. Nor will you find any two employers with the same system setup or skill requirements. No licensing bureau can improve the employee interview process. Only training your managers in management and interview skills can do that.

Instead of presenting this technical forum with well organized facts and logic, Mr. Maginnis has approached this subject in a typical liberal fashion. He has used circular logic, scare tactics, contradiction, me-too ism, and unsubstantiated assertions presented as if they were facts.

Microsoft figured out that the performance of their products could be degraded by poor technical skills. This in turn, hurt the customers' impressions of their products. The “Microsoft Certified Systems Engineer” program was started to cure both of these problems and enhance their sales. But, notice, this was done “within” the industry. And, it was done their way and on their timetable with no government involvement or interference.

I will never willingly support the involvement of any government agency in the programming profession. If major players in Linux see problems similar to Microsoft, maybe they should pursue a similar solution. If you do some research, you will find that there have been previous attempts like this one and all of them have failed. Anyone remember the DPMA's Certified Data Processor?

—Richard McClendon Jr, rhmjr@flash.net

Your Article "Linux Certification" in April 1999 Linux Journal

Dear Dr. Maginnis:

As an MCSE/MCP+I/MCT, I couldn't disagree more with you regarding the "need" for Linux certification.

Every certification program I've seen through the years (A+, CNE/CNA, MCSE, etc.) starts out with the best intentions, but eventually loses its way by getting caught up in the "business of certifying people".

One problem is that no one way of becoming "certified" is the best. One thing that "should" happen is that the companies doing the certifying need to properly screen prospective students to properly guide people with different levels of skill into the appropriate training tracks, & recommend opportunities for gaining hands-on experience so that we don't end up with "paper" MCSEs, CNEs, whatever. My experience as an MCT working with ATECs as a contractor is that they don't do this of course because that path of integrity could cost them money.

As an MCT it amazes me the expectations people have who get sold these certification programs by some sales rep working for an ATEC. They think that last week they were an interior decorator, they went a "Bill Gates Wants You" career night, now they're in this certification program, and in a couple of months with their freshly minted MCSE they'll get $60,000/year starting salary without any experience. Especially pernicious in my mind in this area are the so-called "boot camps" which cram certification preparation into a couple of weeks.

Another issue is that just because a job candidate has an acronym following their name doesn't mean they have the requisite skill level; employers still have a responsibility to screen these people to avoid hiring someone unqualified.

When I was a contractor managing a web development project last year, I consistently told my company to require all new hires to agree to work on 90 day contract-for-hire to prove themselves.

Another problem with cranking out a certification program is the logistics of developing and deploying quality materials. I am phasing out the MCT portion of my training business and focusing on custom training and IBT development, actively turning down offers to teach at ATECs primarily because of having to use Microsoft's Official Curriculum materials, which are of "inconsistent" quality.

Based on my experiences, my advice to the Linux community regarding certification is if you're going to do it, learn from the mistakes of your predecessors and don't let the process turn into a "puppy farm."

Sincerely,

—R Chance Brents, brentorg@home.com

Re: Your Article "Linux Certification" in April 1999 Linux Journal

Dear Mr. Brents,

Thank you for taking the time to comment on my article in the *Linux Journal*. I must say that I am impressed by the depth of your comments, your knowledge of the industry, and your true concern for quality education. Allow me interleave my other comments with yours.

> As an MCSE/MCP+I/MCT, I couldn't disagree more with you regarding the "need" for Linux certification.

I'm not sure what you mean by "need," but I see a need for employers to be able to make some assessment of an applicant. This is a clear trend in the industry and if Linux is to part of the main stream, then there must some form of certification. On the other hand, there no "need" to exploit people. There is no need to take some poor, ill prepared individual and suck the last few dollars out of his or her savings account just so they can have another reason to fail. I strongly disagree with this "need."

> Every certification program I've seen through the years (A+, CNE/CNA, MCSE, etc.) starts out with the best intentions, but eventually loses its way by getting caught up in the "business of certifying people".

As one with an academic background, I do not have the experience in the industry that you have had so I find your comments helpful.

> One problem is that no one way of becoming "certified" is the best.

I think you are saying that students seem to learn independent of how they are taught and, if so, I agree. As teacher with 20 years of experience, I can tell you that I do not (and my peers do not) have any idea what happens inside the heads of students when they learn. But I do know that when I specify a clear criterion of what "knowledge" is, then I can construct tests that are a fair assessment of that knowledge. One of my favorite student survey responses went something like this: "He gives tests on what I know, not what I don't know."

> One thing that "should" happen is that the companies doing the certifying need to properly screen prospective students to properly guide people with different levels of skill into the appropriate training tracks, & recommend opportunities for gaining hands-on experience so that we don't end up with "paper" MCSEs, CNEs, whatever. My experience as an MCT working with ATECs as a contractor is that they don't do this of

> course because that path of integrity could cost them money.

As I understand the process, there are sales people at the bottom continuously calling new prospects and trying to bring new recruits into the training centers. Since the sales people live on commission, there is no telling what has been said to these recruits by the time they sit down for the first class.
But at the same time, these training centers will only teach what they are allowed to teach by the vendor. So it seems that you are saying that the vendors are not providing the proper training materials? Or maybe you are saying that the vendors are just throwing out a static product and not receiving information on the type of student studying the material.

Tell me if I am wrong, but it seems to me that the more material the vendor offers the more opportunity the trainer has to train. So the trainers would welcome more “tracks” to teach.

> As an MCT it amazes me the expectations people have who get sold these certification programs by some sales rep working for an ATEC. They think that last week they were an interior decorator, they went a “Bill Gates Wants You” career night, now they're in this certification program, and in a couple of months with their freshly minted MCSE they'll get $60,000/year starting salary without any experience.

I know what you are describing, I have seen my fair share of students with lots of dreams, low ability, and poor study habits, but I'm lost on your overall point. I understand the sales people have “over sold” the product and that the student comes in with a lot of “hope” but it's also my impression that the six classes that the student must traverse are, in fact, “significant.” I also understand that they do not have the proper experience to back up what they have learned, but have they not learned? On the resume, it may say “MCSE” but the employer will also see that there is no experience to back it up.

> Especially pernicious in my mind in this area are the so-called “boot camps” which cram certification preparation into a couple of weeks.

There is another, more general, name for this and it's called fraud. In my mind, the vendor permits this by using test questions that are few in number, “stale,” and overly narrow.

> Another issue is that just because a job candidate has an acronym following their name doesn't mean they have the requisite skill level; employers still have a responsibility to screen these people to avoid hiring someone unqualified.

In other words, employers are not checking up on past projects and recommendations from others?

> When I was a contractor managing a web development project last year, I consistently told my company to require all new hires to agree to work on 90 day contract-for-hire to prove themselves.

That's even better. But the question is: with other things being equal, did you pick a certificate holder over a non-certificate holder?

> Another problem with cranking out a certification program is the logistics of developing and deploying quality materials. I am phasing out the MCT portion of my training business and focusing on custom training and IBT development, actively turning down offers to teach at ATECs primarily because of having to use Microsoft's Official Curriculum materials, which are of “inconsistent” quality.

Again, you seem to be saying the problem lies with the vendor by not taking the time to properly understand the needs of the client.

> Based on my experiences, my advice to the Linux community regarding certification is if you're going to do it, learn from the mistakes of your predecessors and don't let the process turn into a “puppy farm.”

Sounds great! So if you are serious about this why don't you help us. We got training operations wanting material and we need a program to train-the-trainers. I have no idea where the money is to pay you at this moment, but I do know that we can develop material for you to train. In this way you can use your experience to help create a solid program that will have the right amount of “tracks” and will not suffer from the excesses of sales people.
Sincerely,

—Tobin Maginnis, ptm@sairinc.com

Internationalization

Recently I tried to fax a letter to your subscription department, and had difficulty dialling the number given in the contact information at the front of the journal. I dialled my international carrier, then the country code (281), then the number. Then I thought to myself “Hmm, 281 is a strange country code”, and it dawned on me - it's an area code ! The country code is 1, of course. What else could it be ?

I think that the failure to list any of your contact numbers in international format is embarrasing. A great deal of the Linux community is outside

the US, and indeed the origins of Linux itself is from outside the US. FYI, the format for international listing is +country-area-number or (+country) area-number. Given the theme of the March issue, don't you think you should get with the program ? :-)

Cheers,

Brian Lowe, lowe@mds.rmit.edu.au

Archive Index Issue Table of Contents

Advanced search

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

linux.com

Marjorie Richardson

Issue #62, June 1999

Recently, VA Research bought the linux.com domain and hired Trae McCombs to be the site manager. I talked to Trae on April 1 to find out his plans.

Recently, VA Research bought the linux.com domain and hired Trae McCombs to be the site manager. I talked to Trae on April 1 to find out his plans.

Margie: Tell us how you came to work on the linux.com site.

Trae: I am constantly amazed I have a job at VA Research. I never graduated from college and I spent four years in the military. At one time, I even had a climbing guide service and taught climbing. In March of 1995, I bought my first computer. I ran Windows for about a year and discovered very quickly that I wanted something else. Then I saw a screen shot of Linux—an old version of Enlightenment called FVWM-XPM—and thought, “I wanna run that!” I actually started running Linux because it was pretty!

Margie: Now that's a Linux story I haven't heard before.

Trae: I know! That was September of 1996, and getting started was pretty hard for me. I think I was the first non-geek, non-technical person to use Linux. Other users told me about RPM and configure->**make->**make install. I have no programming skills—I couldn't code to save my life!

Margie: So what are you doing working for VA Research?

Trae: That's an interesting question! They hired me for this position because of what I did for the themes.org site. I talked to all the major window manager authors about setting up an official theme site for their window manager. After getting their approval, I started building this huge themes repository for the Linux community. I couldn't have done it alone—I built a staff of about 18 volunteers who I managed. I think Larry Augustin was impressed with how I managed that site, and that's why he hired me.

For linux.com, I'm using that same model. To develop the web site, I have 13 volunteer staff people, and I'll probably scale up to 20 or 25. It's so easy to get people. This type of community involvement is what drives Linux and the Linux kernel. Linus said, “Hey, I can't do everything on my own. Let's bring in other people to do it.” I am using this same model to build linux.com.

Margie: Tell me about the site.

Trae: linux.com is going to have a broad scope. It is going to be *the* quintessential Linux portal. When people hear about Linux, they will first go to linux.com to look for information. The press will also be able to get various tidbits of information they need. Phil Hughes is on our advisory board—we can't go wrong!

On the home site, there is going to be a huge “What is Linux” button. All around the button will be editorials, interviews, various other content and links to different things. A button bar at the left side will be totally configurable. People will be able to add links to their favorite places to the button bar. No one is doing that—others are just including everyone else's news. To me, this takes away from the experience of actually visiting other people's web sites.

The main feature is the “What is Linux” section that includes three areas: corporate, new to Linux and new to computers. When a visitor clicks on one of these areas, he will be walked through a “power point-esque”-type presentation, as opposed to a text FAQ. It's not going to be a full multimedia experience, but it will be very clear and concise, and at any time, a jump can be made to another path or presentation. The layout the staff is working on is very graphically pleasing, and more is yet to come.

Margie: Other sites do this same sort of thing. What exactly will make linux.com unique?

Trae: We will have a chat section off the main page where people will be able to get tech support from the community. No one does that now. Everyone else offers e-mail or other types of support. We'll have a “Talk with Linux Users” button where it will be easy to find. With just a click, they will be talking to other

Linux users and finding answers to their questions. The only reason I am running Linux right now is because the Linux community is so open-armed and willing to help. They spend hours on IRC—countless hours—helping people like me. That's an invaluable resource.

Margie: Yes, it is. I'm always amazed how many people seem to have the time to do that. So, the chat space is why people will come to this site?

Trae: People will come because we are *linux.com*!

Margie: Oh, because of your name!

Trae: Yes, the name has power, lure—it is the main reason all those other people wanted this domain.

Margie: When do you expect linux.com to be up?

Trae: May 18—the start date for Red Hat's Linux Expo in North Carolina. I wanted to launch it at a trade show, and Linux Expo seemed the right time.

Margie: I'll be at the Expo this year, so I'll see you for the announcement. Any comments about linux.org?

Trae: My hope is that when we bring out this truly awesome site, linux.org will say “Okay, that's great! Let's one-up them.” Then we'll say “Oh, wait a minute—let's one-up what they've done!” Guess who wins this game of “one-up-manship”? The users!

Margie: That is certainly a great attitude. Any closing thoughts?

Trae: As long as the Linux community wins, there will be no fracturing. We will never have hard feelings towards any site. I want to see Linux win. I'd also like to thank the staff and the rest of the Linux community who helped get us here.

Archive Index Issue Table of Contents

Advanced search

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

The Other Shoe

Doc Searls

Issue #62, June 1999

Every vendor wanted to be your only vendor—never mind that no vendor could provide anything close to all the services you needed.

Before the Internet, networks were private affairs. You bought them from Novell or Microsoft or IBM. More accurately, you bought a set of network services you could obtain only from certain companies. You got *file* and *print* from Novell or Microsoft, *messaging* from Lotus, *security* from IBM and so on. None of those services would cooperate with those same services from other companies. That's not how “The System” worked. Every vendor wanted to be your only vendor—never mind that no vendor could provide anything close to all the services you needed.

The Internet broke that system by taking several network services and putting their protocols (i.e., HTTP for Web, SMTP for messaging, FTP for file transport and others) in the public domain, running on top of the equally public network protocols of TCP/IP. This gave us a high degree of interoperability for the first time. The Internet still does not perform many services at all, or performs them incompletely, so it did not put Novell and Microsoft out of business. The “Old Guys” still perform very well, but the Internet gave their businesses a whole new context—a public one. In a blink, their whole world changed, along with everyone else's.

It's still changing. When the Internet finishes settling in, almost everything in communications will depend on it, from embedded controllers to Palm Pilots to the entire phone system. But perhaps the Internet was just the sound of one shoe dropping—and the other shoe is Linux.

Linux seems to have the same calling: something free for everybody. The Internet seems to be its natural companion. Already the vast majority of web pages are served by Apache, running mostly on Linux systems. Are web

services an enterprise beachhead for Linux? Or are they just a vertical application with little leverage?

Is there a big-time business model for any Linux vendor, besides attracting expensive investment bets from Microsoft OEMs and partners looking for alternatives? Are there other possibilities, all as radical and sensible as Linux itself?

How hard is it to imagine Linux kernels as the most basic building blocks of networked life? For Linux geeks, it goes without saying. For the rest of us it's a big stretch, especially when Linux still looks like a Swiss watch delivered in 400 separate parts. But so was imagining a networked world that wasn't a medieval mess of warring states dominated by the Empire of Microsoft, the Duchy of Novell and the Blue Kingdom of IBM.

We're still less than halfway through the shift from personal to social computing. Most households do not have PCs, and most that do are not connected to the Net. According to design critic and user advocate Don Norman, the two basic reasons for this are: computers are too complicated for many people, and the Net still lacks a plug-and-run infrastructure. He lays out a short-form prognosis in the title of his latest book, *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution.*

People are social. Telephony is equally social, because it lets people converse over simple appliances (incomprehensible cell phones and PBXes notwithstanding). Computing is social too, but only for a minority. There is still no computing appliance that's as social as the telephone. Will Linux deliver it?

I suggest that the first social computing appliance—let's call it the first SC—will cost less than $400, look friendlier than an iMac, get on the Internet with the ease of a phone call, and produce Microsoft Office-compatible files for those who want to use simple productivity applications.

Well, a *prototype* of that appliance can be built right now by slapping just about any Linux, any desktop and any office suite onto a commodity PC too slow for Windows 98, and a Windows newbie will probably know how to use it.

On the server side, the appliances are already here. The Cobalt Qube 2 is a seven-inch blue cube that anyone with an IP address can put on the Web in minutes and control easily over a browser. It runs Apache on Linux, but Cobalt doesn't mention that fact anywhere other than its data sheets. Why? Because it's an *appliance*. What's in it matters no more than what makes heat in a steam iron. All that matters is that it works.

Right now, the companies moving in the appliance direction are what economists call “fast followers.” Microsoft was once a fast follower of Apple, which is why Windows is a MacOS knockoff. Now Corel, KDE, Star Division and GNOME are fast followers of Microsoft, which is why their goods are not only effective clones, but in some cases (most notably Corel's) significantly improve on what they copy. Are they headed in the right direction? Not if they don't follow the user even more obsessively than they follow Microsoft. Don Norman says:

> The successful family of information appliances will be built around the people who use them and the tasks to be performed. Products in the world of information technology have suffered far too long under the existing technology-centered designs... Today it is the individual who must conform to the needs of technology. It is time to make technology conform to the needs of people.

Currently, Linux requires far more compliance than Windows or Macintosh. Can that change? Only if some folks in the Linux community start to remember “the rest of us” abandoned by Apple fifteen years ago. If they do, the other shoe is sure to drop—big time.

Archive Index Issue Table of Contents

Advanced search

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

New Products

Ellen M. Dahl

Issue #62, June 1999

PCI-Fibre Channel RAID Controllers, Applix SHELF 1.3, HELIOS Products for Linux and more.

PCI-Fibre Channel RAID Controllers

ICP vortex announced their GDT7519RN (one FC port) and GDT7529RN (two FC ports) controllers which support RAID levels 0, 1, 4, 5 and 10. They feature 64-bit design for very high data transfer rates, including a primary 64-bit PCI interface clocked at 33 MHz, as well as an onboard 64-bit bus for maximum throughput and a 64-bit RISC CPU for heavy I/O operations. The hardware XOR engine is optimized for RAID 4/5 operations. ICP's 64-bit RAID Controllers can be run in either 32-bit or 64-bit PCI slots. Drivers are available for many operating systems including Linux. ICP is committed to supporting current and upcoming versions of Linux with all of its current and upcoming RAID controller lines. Please contact ICP vortex for pricing.

Contact: ICP vortex Corporation, 4857 West Van Buren Street, Phoenix, AZ 85043, Phone: 602-353-0303, Fax: 602-353-0051, E-mail: sales@icp-vortex.com, URL: http://www.icp-vortex.com/.

Applix SHELF 1.3

Applix, Inc. launched its first Open Source initiative with SHELF, an embeddable, full-featured, graphical programming language. Both Applixware and Applix SHELF are available for all major distributions of Linux, as well as other

platforms. Applix SHELF is released under the GNU Library Public License as defined by the Free Software Foundation. Version 1.3 and 1.1 are available for free download at Applix's web site.

Contact: Applix, Inc., 112 Turnpike Road, Westboro, MA 01581, Phone: 508-870-0300, E-mail: info@applixware.org, URL: http://www.applixware.org/.

HELIOS Products for Linux

HELIOS Software GmbH announced the availability of its EtherShare 2.5, EtherShare OPI 2.0, PDF Handshake and Print Preview products for the Linux operating system on computers based on Pentium processors. HELIOS products run on powerful and scalable servers from Apple Computer, Data General, Compaq (Digital), HP, IBM, Intel (Pentium), Motorola, SGI and Sun, providing reliable cross-platform support for Macintosh, DOS/Windows and UNIX-based clients. Prices in Euros are available on the HELIOS web site.

Contact: HELIOS Software GmbH, Steinriede 3, D-30827 Garbsen, Germany, Phone: +49-5131-709320, Fax: +49-5131-709325, E-mail: marketing@helios.de, URL: http://www.helios.de/.

Enhydra

Lutris Technologies announced its new Open Source Java application server development environment for Linux, suitable for building and deploying high-grade web applications. Features include the Enhydra Application Framework, a super-servlet run-time environment of common services for supporting N-tier Enhydra applications; Enhydra Multiserver, a servlet-running environment with graphical servlet management, monitoring and debugging; Enhydra XMLC, an XML compiler designed to support designer and Java developer co-development; Enhydra JDDI, a structured approach to using embedded Java for dynamic HTML; and Enhydra DODS, a graphical tool for object-to-relational database mapping. The product is free, with a FreeBSD-style license and Linux RPM installation support.

Contact: Lutris Technologies, 1200 Pacific Avenue, Suite 200, Santa Cruz, CA 95060, Phone: 831-471-9753, Fax: 831-471-9754, E-mail: info@enhydra.org, URL: http://www.enhydra.org/.

LDAP Proxy Server 2.0

Innosoft International, Inc. announced enhancements to its Innosoft LDAP Proxy Server (ILPS) to provide automatic load balancing among multiple LDAP servers and transparent failover for high-availability directory services. Innosoft Directory Service (IDS) products are available for Red Hat Linux 5.1 on Intel. Prices for the Innosoft LDAP Proxy Server begin at $4,800 US for up to 15 concurrent connections.

Contact: Innosoft International, Inc., 1050 Lakes Drive, West Covina, CA 91790, Phone: 626-919-3600, Fax: 626-919-3614, E-mail: sales@innosoft.com, URL: http://www.innosoft.com/.

DeveloperX2 2.1

NetBeans introduced DeveloperX2 2.1, one of the first full-featured, cross-platform Integrated Development Environments (IDEs) to support and run on Sun Microsystems, Inc.'s Java 2 platform. It enables software developers to build sophisticated Java Foundation Classes (JFC) GUIs, compile and debug applications on the platform of their choice. NetBeans also launched a concurrent version, Developer 2.1, that supports Swing 1.1 and Java Development Kit (JDK) 1.1. The license for all versions of Developer 2. *x* is $145 US for one to four users. Developer runs on all platforms that support JDK 1.1 and 1.2, including Linux.

Contact: NetBeans, Inc., Pod Hajkem 1, 180 00 Prague 8, Czech Republic, Phone: +420-2-8300-7322, Fax: +420-2-8300-7399, E-mail: info@netbeans.com, URL: http://www.netbeans.com/.

System Manager in a Box

PegaSoft announced the beta release of their new Linux product, System Manager in a Box (SMiaB). This administration tool addresses the need for experienced system administrators to diagnose installation, configuration and run-time problems. Using artificial intelligence techniques, SMiaB performs more than 2000 system checks and reports not only which files have problems, but which systems are affected and why. SMiaB runs on all major flavors of Linux. Scheduled for release in late spring 1999, SMiaB 1.0 will include commercial editions geared especially to business. The beta is available for download from the PegaSoft web site.

Contact: PegaSoft Canada, 2631 Honsberger Avenue, Jordan Station, ON L0R 1S0, Canada, E-mail: pegasoft@tiamet.vaxxine.com, URL: http://www.vaxxine.com/pegasoft/.

ResQ!Net and ResQ!Net for IBM's Host-On-Demand

ResQNet.com, Inc. announced complete Linux support with its flagship products, ResQ!Net and ResQ!Net for IBM's Host-On-Demand 99. The ResQ!Net products provide graphical 3270/5250 host connectivity for Linux workstation and server applications by instantly transforming old-fashioned mainframe and IBM AS/400 screens into modern graphical interfaces with an easy-to-use Windows look and feel. It is available as both a stand-alone application and as an add-on to IBM's eNetwork Host-On-Demand. ResQ!Net is usable out of the box with existing Linux applications. A free trial version may be requested via the web site.

Contact: ResQNet.com, Inc.,130 Cedar Street, 5th Floor, New York, NY 10038, Phone: 212-537-4800, Fax: 212-294-8822, URL: http://www.resqnet.com/.

SNAPIX 3.1.18

ADNT announced a new version of SNAPIX Special Edition for Linux (SEL), an X11/Motif Interface and application generator. It provides an object-oriented, event-driven language, allowing easy use of X11 and Motif libraries under UNIX. SNAPIX SEL is downloadable for free from ADNT's web site, complete with statically linked Motif libraries, so no royalties are due OSF. Commercial versions of SNAPIX are available for Linux and other platforms for those users with a Motif license.

Contact: ADNT, 4 rue Louis Massotte, 78530 Buc, France, Phone: +33-0-1-3920-1010, Fax: +33-0-1-3956-5592, E-mail: info@adnt.fr, URL: http://www.adnt.fr/.

Sophos Anti-Virus for Linux

Sophos Inc., a developer of network-oriented anti-virus software, announced the release of Sophos Anti-Virus for six versions of UNIX, including Linux. The UNIX implementation of Sophos Anti-Virus includes Sophos' unique SAVI API, enabling third-party developers to integrate with the Sophos Anti-Virus interface. An evaluation copy is free; visit the web site for downloading and pricing of the full product.

Contact: Sophos Inc., 18 Commerce Way, Woburn, MA 01801, Phone: 888-SOPHOS-9, 781-932-0222, Fax: 781-932-0251, E-mail: sales@sophos.com, URL: http://www.sophos.com/.

WebFOCUS

Information Builders announced Linux support for the web-enabled enterprise analysis and reporting solution, WebFOCUS. The WebFOCUS Managed Reporting Environment lets users with little or no database experience create sophisticated reports using objects that have been preconstructed by a database administrator. The WebFOCUS Suite includes Report Broker, an intelligent, Java-based engine that centralizes the scheduling and distribution of reports over the Web, e-mail, printers and faxes. WebFOCUS is available to channel partners through Information Builders' InfoElite Partners Program. Pricing is to be determined at country of origin.

Contact: Information Builders, Two Penn Plaza, New York, NY 10121-2898, Phone: 212-736-4433, Fax: 212-967-6406, E-mail: askinfo@ibi.com, URL: http://www.ibi.com/.

PC ParaChute

UniTrends Software Corporation announced PC ParaChute, a complete crash recovery product for Intel-based PCs. PC ParaChute backs up each PC on the network to the central server through a TCP/IP connection. PC ParaChute can create customized bootable diskettes either centrally or remotely for each PC. It offers such technology as bit-level verification of the backup, software compression (to reduce network traffic), DynaScan-backup verification weeks to months later and a free-space optimizer. PC ParaChute requires Backup Professional, also available for Linux. The cost of PC ParaChute is $35 US per PC.

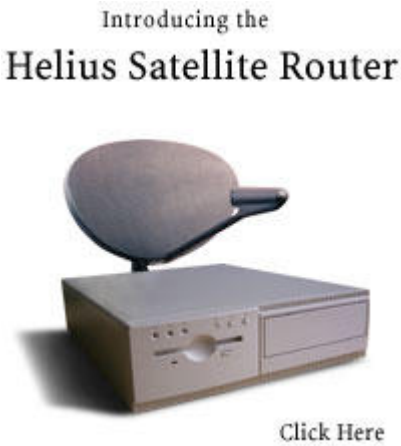
Contact: UniTrends Software Corporation, 1601 Oak Street, Suite 201, Myrtle Beach, SC 29577, Phone: 800-648-2827, Fax: 843-626-5202, E-mail: sales@unitrends.com, URL: http://www.unitrends.com/.

Web+ 4.0

TalentSoft announced its newest version of Web+, a development tool dedicated to creating web-based client/server applications without writing CGI programs. Web+ enables the creation of highly functional web pages that integrate with databases, file systems, e-mail, Java applets, legacy applications and other TCP/IP applications using socket technology. In addition to supporting ODBC database connectivity, Web+ 4.0 now has built-in native API connectivity with mySQL, miniSQL and PostgreSQL databases. Prices and licensing begin at $100 US for the Developer's Edition with two concurrent connections.

Contact: TalentSoft/Talent Information Management, LLC, 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/.

Helius Satellite Router



Helius, Inc. released the Helius Satellite Router, the first solution for small- to medium-sized businesses and K-12 schools needing high-speed Internet access and support from any location. The Satellite Router is an all-in-one box that includes Virtual Technician and Helius Optimized software. The Helius Satellite Router is a thin server network appliance that works with any local area network, including Linux. System capabilities include interactive Internet, IP Multicasting, e-mail, news, Web, AppleTalk and SMB servers, and caching and proxy services. Installation, setup and support are easy and worry-free with Helius Virtual Technician. Prices start at $2,499 US for up to 30 concurrent users.

Contact: Helius, Inc., 240 West Center, Orem, UT 84057, Phone: 888-764-9020, 801-764-9020, Fax: 801-764-9022, E-mail: info@helius.com, URL: http://www.helius.com/.

Archive Index Issue Table of Contents

Advanced search

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

Best of Technical Support

Various

Issue #62, June 1999

Our experts answer your technical questions.

Correction

I was reading the BTS column in the April issue, and noticed that for the “Wrong Date” question from Bilal Iqbal, you edited my answer, changing the meaning. In fact, you reversed the arguments to the **ln** command. The link should be

```
ln -sf /usr/share/zoneinfo/US/Pacific \
   /etc/localtime
```

and the **-f** option is most likely needed, since most systems already have a link there. Also, I said “a link like this”, not “create this link”, since the reader specified his timezone was GMT+5, so telling him to create a link setting his timezone to GMT-8 isn't exactly what he would want to do. —Marc Merlin, marc@merlins.org

Zip Drive Affects Printer

I am using a Zip drive with Red Hat 5.2 and cannot use my printer because the Zip drive is a parallel port version. The printer manager does not recognize the printer is connected. I was able to use the printer before the Zip drive was installed. Is it possible to use both the zip and the printer? I know I cannot use it while my zip is mounted, but when I unmount the drive, would it be possible? —Smileyq, smileyq0@mindspring.com

Rebuild the kernel, defining lp and zip support as modules. When you wish to use the printer, unload the zip module (if loaded) and load the lp module, so you can use the printer. When you wish to use the zip drive, unload the lp module (if loaded) and load the zip module. That's it. —Paulo Wollny, paulo@wollny.com.br

According to the Zip Drive mini HOWTO, question 7.1 (metalab.unc.edu/LDP/HOWTO/mini/ZIP-Drive-7.html#ss7.1), it should be possible. You will need a newer kernel (2.2.x) or you'll need to upgrade the ppa driver in your current kernel source and recompile it. Since RH 5.2 isn't fully compatible with the 2.2.x kernels, you may be better off recompiling your current kernel, and you can find the ppa driver on David Campbell's page: http://www.torque.net/~campbell/. —Marc Merlin, marc@merlins.org

Restoring DOS Data on Hard Drive

I just got Linux about a month ago (Red Hat 5.2) and have been experimenting with it. Last night I accidentally ran **mkfs.msdos** on my Windows 98 FAT32 partition (/dev/hda1) thinking it was a command to mount an MS-DOS partition under Linux, but of course, it nuked my drive, created an MS-DOS partition over it, and I lost everything on my drive! I am writing to you in hopes that you know of a way that I can salvage the information still on my drive. The mkfs.msdos command erases only the FAT sector when it creates a new file system, right? So, shouldn't all the information still be there? Thank you in advance for any assistance. —Jon Verville, theverv@hotmail.com

That's a rough accident. The short answer is there's little you can do. Yes, the information is still there, but the FAT tables tell the system where to look for different pieces of a file, and if your file system is fragmented, it could be very difficult to recover anything.

However, it would be worth trying a few recovery tools, such as Norton's Disk Doctor if only to salvage some of your data before you reinstall Windows. You may be able to save something from your disk if you touch —Chad Robinson, chad.robinson@brt.com

Linux vs. IRIX

Normally, I am an IRIX user. Recently, I bought a dual-Pentium machine and installed Linux SuSE 5.3. on it. I can't figure out if my second processor is recognized; there seems to be no command like **hinv** in IRIX. Any suggestions? Is there any document comparing IRIX and Linux commands? —Tobias Knaute, tobias.knaute@charite.de

First, you have to make sure your kernel is compiled with SMP support; this is not the default for most distributions. Then check your /proc/cpuinfo file which contains the information for all CPUs found during boot time.

In order to take full advantage of your dual processor machine, I'd suggest you use the 2.2.4 kernel version, which is the latest at this time. —Mario Bittencourt, mneto@buriti.com

Removing xeyes

I installed xeyes in the KDE menu and while trying to remove it, it multiplied. Is there any way to close it? It has no resize window and every method of stopping or interrupting doesn't prevent it from returning on bootup. I have searched the man pages for a key combination to kill them to no avail —Edward Spadacene, espada@mbox.kyoto-inet.or.jp

You can close xeyes by clicking the right mouse button on it and choosing "Close" from the pop-up menu that appears. Next time you start KDE, xeyes will not be run. —Scott Maxwell, s-max@pacbell.net

Setting Up for Games

I am trying to set up my firewall so that my users can play on-line games. In particular, I need to set up the following ports:

- An initial outbound TCP connection on port 47624
- Subsequent connections of inbound and outbound TCP and UDP ports 2300-2400

I am using IP masquerading. My firewall is an ipforwarding one, i.e., not a proxy firewall. Any help you can give me would be greatly appreciated. —Neil Shanks, neilshanks@home.com

Unless you set up additional firewalling rules, there is no way to allow the outbound packet on port 47624. Outbound UDP and TCP connections in the 2300-2400 range will work fine, and the masquerading machine will open a reverse connection to gateway back inbound packets if they come back on the same port. If they don't, you may have some luck with the **ipautofw** packet forwarder which you can get at ftp://ftp.funet.fi/pub/Linux/PEOPLE/Linus/net-source/firewall/ipautofw.tar.gz. You may also want to look at the ipmasq-HOWTO and the list of applications that can be made to work through IP masquerading, http://users.nais.com/~nevo/masq/. —Marc Merlin, marc@merlins.org

Downloading with Netscape

Is there a default directory for downloads? I used the included Netscape Communicator v4.07 to download the Corel WordPerfect Suite 8 for Linux (a 25MB download that took a couple hours) and the Quake 2 for Linux demo. When I went looking for the files to install them, I couldn't find them anywhere. The system did not ask for a specific location to place them, so I assumed there was a default location. Am I wrong? —Robert Gray, noeman5@hotmail.com

By default, Netscape tries to save the file in the user's home directory or the last place (path) where you saved a file. To locate the files, use the command **find**:

```
find / -name "corel*.tgz" -print
```

—Mario Bittencourt, mneto@buriti.com

More on Recovering Data

I had a hard disk crash and cannot mount root directory /dev/hda3. Is there any way to perform a partial recovery of the data on the drive or split the root partition (skipping the bad sectors) from a boot floppy? I only need the most recent data in the mail directory that was not backed up. —Tom Voydanoff, tvoydan@systechcorp.com

Boot using a rescue disk and try to repair the partition with the command **fsck.ext2 /dev/hda3**. After that completes, try to mount the partition yourself and dig out the needed files. —Mario Bittencourt, mneto@buriti.com

Errors in Hard Drive

I have a Linux system that the hard drive died on. I have a tape backup of the entire system. I did a minimum install of Red Hat to get the new drive running, created the partitions / and /big and told **taper** to overwrite all files. It ran and restored its files but had over 1200 errors. I viewed the log, which contained statements about checksums not really errors. I was hoping to be all right, but when I rebooted, the system just printed **LI** and stopped. Any help you could offer would be greatly welcomed. —Jabe Pitts, Jabe.Pitts@cwix.com

Your system is probably fine—this is a boot loader issue. Boot loaders need to know where the kernel is located, and by restoring your tape, you overwrote the kernel, which moved it to a new position on the drive. Use a boot floppy to boot your system and run **lilo**. This will reinstall the boot loader. (Note that you may need to use —Chad Robinson, chad.robinson@brt.com

Installing guile

I got a message like the one below when I was trying to install **guile**. I usually can figure out what needs to be installed when I get these failed dependency messages, but this time I'm flummoxed. I have both glibc 2.0.7 and 2.1.x installed. Is this just a peculiarity of the RPM? I've gotten this message with a bunch of RPMs I've tried to install recently, not just this one.

```
[root@localhost new]# rpm -Uvh guile-1.3-4.i386.rpm
failed dependencies:
libdl.so.2(GLIBC_2.1) is needed by guile-1.3-4
```

```
libdl.so.2(GLIBC_2.0) is needed by guile-1.3-4
libm.so.6(GLIBC_2.1) is needed by guile-1.3-4
...
```

—Brady Hegberg, bradyh@bitstream.net

Well, it turns out that glibc 2.1 isn't exactly binary compatible with glibc 2.0. It looks like that's the problem you're having. Grab the .src.rpm file instead, and do

```
rpm --rebuild guile-1.3-4.src.rpm.
```

The resulting .i386.rpm should work and will be in the /usr/src/redhat/RPMS/i386/ directory. —Marc Merlin, marc@merlins.org

Remounting Disks

I am a Linux newbie. Whenever I accidentally shut down my system without halting, I get a message telling me to run disk utilities, and remount hdd. I was wondering, what disk utilities? How do I run them? How do I remount disks? —VoodooXpert, thaiguy@uswest.net

The utility to run is **fsck** which checks and repairs a Linux file system (whether it is ext, ext2, etc.). In order to mount one yourself, type:

```
mount -t filesystem-type device mount-point
```

For example:

```
mount -t ext2 /dev/hda3 /archive
```

Just make sure you already have the mount point created (use **mkdir**), and you are using the correct file system type (ext, ext2, vfat, etc.).—Mario Bittencourt, mneto@buriti.com Many on-line help resources are available on the SSC web pages. Sunsite mirror sites, FAQs and HOWTOs can all be found at http://www.linuxresources.com/.

Archive Index Issue Table of Contents

Advanced search

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

Product Review: Pro-Lite Scrolling Message Signs

Walter Stoneburner

Issue #62, June 1999

A review of the Pro-Lite Tru-Color II PL-M2014R, an affordable multi-color LED scrolling message board that is capable of being controlled by a standard RS-232 serial port.

- Manufacturer: Pro-Lite
- E-mail: info@prolite.com
- URL: http://www.pro-lite.com/
- Price: $150 US
- Author: Walt Stoneburner

The Pro-Lite Tru-Color II PL-M2014R is an affordable multi-color LED scrolling message board that is capable of being controlled by a standard RS-232 serial port. The sign is obtainable from Pro-Lite directly (http://www.pro-lite.com/), but can also be purchased at various discount warehouses for approximately $150. The serial cable and Windows software are sold separately.

This article is not just a review; it can serve as a primer for Pro-Lite's PL-M2014R with ROM release 5.24Q and 32K of memory with Trivia mode, basically your standard sign. Until now, not much developer information has been available to the public, meaning signs were usually configured with the included infrared remote control to display static messages. For very little money, it is possible to build your own serial cable and control the sign using Linux to display more than static text.

The business and personal applications for a highly visible sign are almost limitless: reporting days until software delivery, announcing traffic congestion, providing the weather, showing the date and time, sending public messages,

reporting system load, announcing new mail, showing who's logged in, warning when disk space gets low, login information, announcing unexpected server outages as a watchdog, etc.

Communicating with the sign is almost as simple as beaming text out the serial port; however, a bit of text manipulation is necessary in order to get the sign to respond and do advanced tasks. Linux handles the serial port communication, so sending information to the sign appears as trivial as writing to a file.

Luckily, the majority of the work can be accomplished by simple scripts. All you need is a basic understanding of the shell, AWK, PERL or Python, and you can be running in almost no time. The first hurdle is to build a cable and configure Linux to talk out the serial port.

Cable Construction

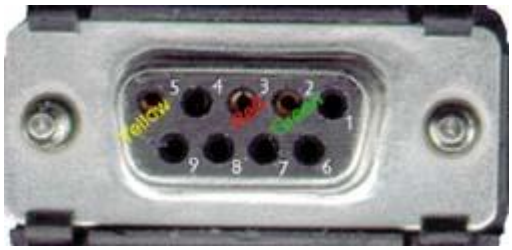


Figure 1. Female DB9 Adapter

The first step is wiring an RJ12 to female DB9 adapter. It requires no tools, and an adapter kit can be purchased at most computer supply stores for about $2 US. The only other thing you need is a length of RJ12 cable with male adapters at each end. This means a standard telephone cord will do nicely. Your cable will most likely have the standard colors black, red, green and yellow, in that order.

Cables come in two flavors: straight-thru and reversed. You'll need to take the RJ12 cable and put it end to end (like a loop) to find out which kind of adapter you need to build.

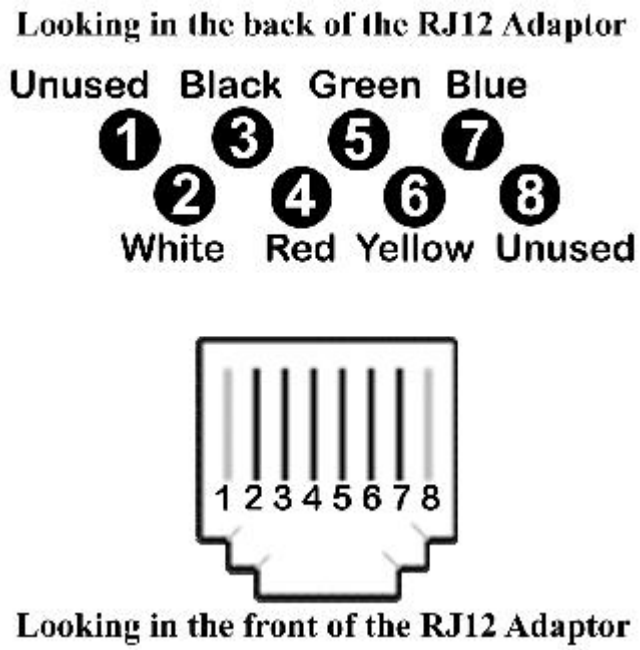


Figure 2. RJ12 Adapter Diagram

One of two things will be noticeable: either the wires will match black to black, red to red and so on, or they will reverse their order showing black to yellow, red to green and so on.

If the cable is a straight-through, where the colors match, wire your RJ12 to a DB9 adapter using Pin 2 as green, Pin 3 as red and Pin 5 as yellow. If the cable is reversed, then where the colors reverse sequence, wire your RJ12 to a DB9 adapter using Pin 2 as red, Pin 3 as green and Pin 5 as black.

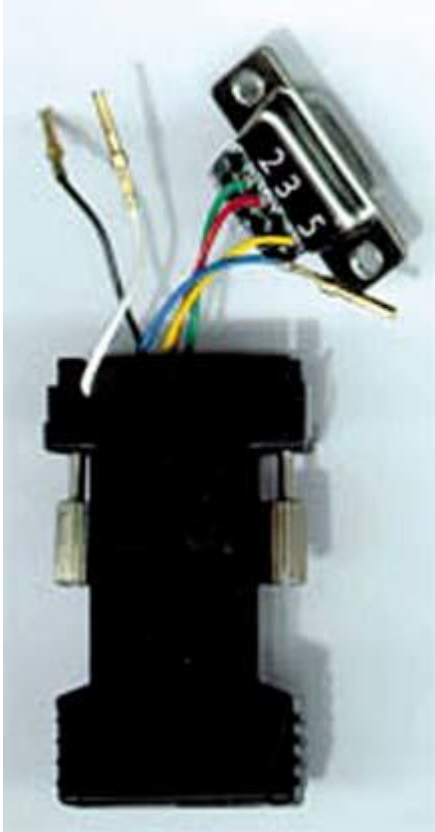


Figure 3. RJ12 to DB9 Wiring

Shove the unused adapter wires into the casing and snap the adapter shut. Take care not to let the exposed ends touch anything metal inside the adapter casing. You may want to clip the unused wires. Put an extension cable on your serial port and connect the adapter wires directly to the extension cable in order to test the wiring configuration before pushing the pins into the connector.

When all is said and done, one end of the RJ12 goes into the side of the LED sign, the other into the adapter you just made, and the adapter plugs into the computer.

Configuring Linux to Talk to the Sign

My sign is plugged into COM1, also known as /dev/ttyS0. I've elected to use a symbolic link to the sign, in the event I ever decide to change to another serial port in the future. To make the link, as root type:

```
ln -s /dev/ttyS0 /dev/prolite
```

I tend to shy away from doing development as root. Putting security issues aside for the moment, we can make the device world-writable by typing:

```
chmod a+rw /dev/prolite
```

The sign communicates using No Parity, 8 Bits, 1 Stop Bit; no handshaking of any kind (hardware or software) is used. Early versions of the sign work only at 300 baud, but they can be upgraded to 9600 baud. All signs I've encountered had the 9600 capability right out of the box. The bottom line is that all signs are capable of communication, and even at 300 baud you can outrun the sign. The only drawback is that the sign's baud rate has to be set by the remote control, according to the setup in the manual. This needs to be done only once.

In theory, the sign requires a 15ms delay between each character sent to the sign. I've found that Linux's device driver seems to work just fine without having to do anything special in the software.

```
stty speed 9600 cs8 -parenb -cstopb cread \
   -clocal -crtscts -ignpar -echo nl1 cr3 < \
   /dev/prolite
```

Naturally, you can substitute any baud rate the sign will handle for the 9600. This command will work when you aren't root because it is world-writable.

The most important piece of information for communicating with the sign is that each command sent to the sign must end with a carriage return/newline pair. This is **ctrl-M ctrl-J** on the keyboard or 0x0C 0x0A in hexadecimal. C programmers will recognize it as \r\n. If you want Linux to handle the end-of-line sequence for you, type the command:

```
stty opost -ocrnl onlcr < /dev/prolite
```

When you send a newline, Linux will send both carriage return/newline automatically. Text can now be listed or redirected to /dev/prolite from the shell. If you want to send the carriage returns yourself, type:

```
stty -opost -ocrnl -onlcr < /dev/prolite
```

Where the options are defined as:

- **opost**: postprocess the output stream.
- **-opost**: do not postprocess the output stream.
- **-ocrnl**: do not convert carriage returns into newlines.
- **-onlcr**: do not translate each newline into a carriage return/newline pair.
- **onlcr**: translate each newline into a carriage return/newline pair.

The **stty** command allows for a shorthand representation of all the termios structures that define the device characteristics. For 9600,N,8,1 with automatic carriage returns, type on one line:

```
stty 0:705:bd:0:3:1c:7f:15:4:0:1:0:11:13:1a:0:
12:f:17:16:0:0:73 < /dev/prolite
```

For 9600,N,8,1 with no automatic carriage returns, use the line:

```
stty 0:700:bd:0:3:1c:7f:15:4:0:1:0:11:13:1a:0:
12:f:17:16:0:0:73 < /dev/prolite
```

Command Syntax

Only a minimal understanding is needed in order to operate the sign. Mastery requires becoming familiar with the protocol and doing a little experimenting.

Multiple signs can be connected to a computer over the same serial port by using a simple telephone-line-splitting Y-connector. Each sign has an assignable logical address that allows you to send messages to particular signs. A logical address is represented in hexadecimal as a number between 01 and FF. Addresses may be shared enabling grouping; in this way, a message to ID01 goes to all signs with ID01.

Communication with the sign is always done with readable ASCII characters; thus, an arbitrary ID of 2F would be designated with the digit “2” and the capital letter “F”.

ID00 is reserved to mean “broadcast to all signs”. By default, a sign is preconfigured as ID01; this can be changed with the remote control. If you are addressing a single sign connected straight to the computer, it will respond with its ID number after each successful command. Messages sent to ID00 do not return a response, and neither does an individual sign when the line is shared via the Y-connector.

All messages to the sign, except for those setting the date and time, are sent in the following format followed by the carriage return/line feed: **<ID*xx*>*command***, *xx* is 00 to FF. Any command longer than 1,023 bytes will be ignored by the sign.

To set the date and time, no ID is needed. The format is ***<TYYMMDDwhhmmss>*** where *YY* is the year, *MM* the month, *DD* the day, *w* the weekday (0=Sun, 1=Mon,...6=Sat), *hh* the hour, *mm* the minute and *ss* the seconds. This can be accomplished with the shell very easily. Set the date/time of all signs with the command:

```
ate "+<T%y%m%d%w%H%M%S>" te "+<T%y%m%d%w%H%M%S>" > /dev/prolite> /dev/prolite
```

The leading **+** sign tells the date command to build a string via substitution; see your man pages for details.

To signal a sign to start listening to responses, send it an empty command with the format **<ID*xx*>**; e.g., type:

```
echo "<ID01>" > /dev/prolite
```

Pages

The most fundamental concept of the sign is the idea of a page. A page consists of a message the sign is to display either now or some time in the future. Pages may contain text, numerics, symbols, font attribute tags, color tags, graphic tags and effect tags. There are 26 pages named, appropriately enough, A to Z. Case does matter when identifying a page.

Any given page can hold approximately 1,012 bytes of information. I say approximately because special tags (see sidebar) consume more than one byte, which means less space. Also, using some trickery by omitting the page directive completely and defaulting to page A squeezes out an additional two bytes, which means more space.

The command to set a page is **<P*x*>**, where *x* is the page name. Any text after this sequence is considered text for the sign. To set the message “Linux Rules” on page A, type the following command:

```
echo "<ID01><PA>Linux Rules" > /dev/prolite
```

It is important to leave extra space at the end so the end of the message is separated from the front of the message as it scrolls.

To delete a page, use **<DP*x*>**, where *x* is the page name. For example, to delete page B, type:

```
echo "<ID01><DPB>" > /dev/prolite
```

Displaying Pages

The sign is constantly displaying a page. If the default page A is scrolling by, then when its content is changed it will immediately start displaying the new message. Otherwise, we must tell the sign which page to run using the **<RP*x*>** command, where *x* is the page to display. Normally, the sign can run only one page at a time.

Once a page has been defined, it can be run. For example, to repeatedly display our message, type:

```
echo "<ID01><RPA>" > /dev/prolite
```

Running an undefined page will result in displaying the sign's demo.

Two points of interest for script writers:

1. It is possible to update the contents of a page not being displayed, then switch to that page at a later time.
2. It is possible to update the currently displayed message. The only problem is the display will be interrupted mid-message.

Sadly, you cannot ask the sign what the contents of a page are, what page it is currently displaying, or when it has started or ended a display sequence. Thus, techniques like double-buffering don't work for continuous messages.

I have received one terse message from Pro-Lite that alluded to a future version of the sign which is designed to address such needs explicitly for computer users.

Timers

In order to display one or more messages at a time, the sign includes ten timers named A to J. Each timer specifies a time and a series of 1 to 32 pages to sequence through. You may repeat pages in a sequence. The time a sequence is displayed consists of a weekday, an hour and a minute, any of which may be wildcarded (*) to match “all”.

When the first timer is defined, the sign is put into “timer” mode and will display that timer's messages immediately. Should two or more timers be defined, the sign seems to wait one minute, then checks each timer to see if one triggered based on the current time. If so, at the end of the displaying message, the new timer's sequence goes into effect. If no rule matches, no change is made.

Note the sign checks the rules for the immediate time. It will not go back to try to find a previous timer. So if you set a timer for 4:15PM and it is now 4:17PM, you've missed the moment when the sign would change.

Should two or more timers both be valid for the current time, the sign unpredictably selects one for display. This means you cannot set one timer to display at 4:15 and another timer to display at 15 minutes past any hour. Both rules would be triggered at 4:15 and it is a toss-up as to which message will be displayed.

The timer command is defined as **<T*x*>*dhhmm*ABC**..., where *x* is the timer, *d* is 0 (Sunday) to 6 (Saturday) for the day, *hh* is a two-digit hour and *mm* is the minute. “ABC...” is a list of 1 to 32 pages to display in the order specified. Note that Page-A and Timer-A are two different, unrelated entities. A timer set for “*****” will be triggered immediately.

The following sequence will display a series of messages at 8:00AM, noon, 1:00PM and 5:00PM:

```
$ cat > /dev/prolite
<ID01>
<ID01><PA>Good morning.
<ID01><PB>Have a nice lunch.
<ID01><PC>Get back to work.
<ID01><PD>Have a safe drive home.
<ID01><TA>*0800A
<ID01><TB>*1200B
<ID01><TC>*1300C
<ID01><TD>*1700D
ctrl-D
```

In order to display a whole sequences of pages, just list more page letters. For instance, it is useful to make a scheme where you assign page letters to different message content. For example, page A could be hourly announcements, page R the runtime status of your machine, page M the message of the day, page T the time and so on. To display several pages right now (including repeats):

```
$ cat > /dev/prolite
<ID01>
<ID01><TA>*****TAMARA
ctrl-D
```

Two interesting tidbits of information:

- For some reason, the specific time of Sunday at midnight (00000 as a timer value) does not seem to work consistently. I suspect it confuses this value with unset timers.
- If a run-page command (**<ID01><RPA>**) is issued, the sign is taken out of “timer” mode and displays the requested page. If you then issue the command **<ID01><RP*>**, the sign will go back into “timer” mode, displaying the last sequence of messages.

To delete a timer, use **<DT*x*>**, where *x* is the timer letter or * for all timers. For example, to delete timer D type:

```
echo "<ID01><DTx>" > /dev/prolite
```

Graphics

There are 26 graphic blocks that can be redefined and are commonly used for graphics. Graphics are inserted into the text via the tag **<B*x*>**, where *x* is a letter from A to Z. For example, this command will display a mug and a wine glass:

```
echo "<ID01><PA><BW> Party Tonight <BZ>"
>\
   /dev/prolite
```

Altering the graphics requires the **<G*x*>** command followed by a string of 126 characters made up of R (for red), Y (for yellow), G (for green) and B (for black or unlit). A sequence of seven rows of eighteen LEDs is specified back to back on a single command line. (See Figure 4.)

Figure 4. A Graphics Block Specification

I know at least one user who loads a font into a sequence of graphic blocks, and in this way is able to display more characters than the sign technically allows by default—very clever.

To delete a graphic block, restoring it to the default, use **<DG*x*>**, where *x* is a letter A to Z; * is a wildcard indicating all are to be deleted.

Trivia

One additional feature of the sign is to display in sequence a trivia question, your message, the answer to the question and your message again. It works its way though a canned list of questions and then starts over. The sign can be loaded with your own list, which does not have to be trivia. The list can be up to 16KB long, leaving 16KB for the pages. If no list is loaded (i.e., you deleted the trivia), the full 32KB of the sign is available for page content. To make your own list, do the following:

```
$ cat > /dev/prolite
<ID01>
<ID01><Q+>
<ID01>What operating system isn't a pig?
<ID01>Linux.
<ID01>What operating system is free?
<ID01>Linux.
<ID01><Q->
Control-D
```

To delete the trivia, just omit the lines between **<Q+>** and **<Q->**. If no messages are loaded, trivia mode is turned off. If they are, trivia is turned on.

Other Useful Commands and Tricks

To reset the sign, removing all timers and pages, use **<D*>** like this:

```
echo "<ID01><D*>" > /dev/prolite
```

Technically, there is no way to bring the sign to a halt. However, you can take advantage of a quirk in the software to do the same thing. If you use the **<FX>** tag in a message to control the speed of the sign, but don't supply a message, the old text isn't cleared and the sign stops scrolling until it gets a new message.

Function Tags

Additional tags are shown in “Function Tabs”. European tags are available for characters shown in Figure 5.

European Table

0	Ă	B	É	T	Ó	g	ù
1	Â	C	Ê	U	Ô	h	û
2	Á	D	È	V	Ò	i	ü
3	À	E	Ë	W	Ö	j	Ÿ
4	Ä	F	é	X	Õ	k	Ý
5	Å	G	ê	Y	Ø	l	ÿ
6	Æ	H	è	Z	ó	m	ý
7	ã	I	ë	[	ô	n	ŭ
8	â	J	Í	\	ò	o	f
9	á	K	Ì	]	ö	p	£
:	à	L	Î	^	õ	q	¥
;	ä	M	Ï	-	ø	r	ὰ
<	å	N	í	a	þ	s	ß
=	æ	O	ì	b	Ú	t	Π
>	ß	P	î	c	Ù	u	ø
?	Ç	Q	ï	d	Û	v	∑
@	ç	R	Ň	e	Ü	w	Ω
A	Đ	S	ň	f	ú		

Figure 5. European Characters Available

Uses for the Sign

It doesn't take much to make the sign useful. For example, scripts that send commands to the sign can be executed every few minutes by editing your crontab file (**crontab -e**) to include:

```
*/5 * * * * /usr/local/bin/sign 1> /dev/null \
2> /dev/null
```

Every five minutes, the system will call /usr/local/bin/sign. In this script, we can place any number of tasks. To show our uptime, add these lines:

```
#!/bin/bash
stty 0:705:bd:0:3:1c:7f:15:4:0:1:0:11:13:1a:0:12\
:f:17:16:0:0:73 < /dev/prolite
echo "<ID01><PU>'uptime' " > /dev/prolite
echo "<ID01><TA>*****U" > /dev/prolite
```

If root wants to see the last line of the log file scroll by, this command will suffice:

```
tail -f -n 1 /var/log/messages |
   awk '{ print "<ID01><PA>" $0 " "; }' >
   /dev/prolite &
```

This job sits in the background, getting lines from the end of the log as they come in, prefixes a **<ID01><PA>** to it and sends it to the sign. The process then goes to sleep until another log entry is made. The only reason to run as root is to get access to the messages file.

The sign can also act as a watchdog for our system by setting up a timer to go off several minutes later. Ideally, the sign will get updated before the timer goes off and the timer will again be set for some time in the future. In the event the sign is not updated, the timer trips and an alternate alert message is displayed.

```
!/bin/bash
stty 0:705:bd:0:3:1c:7f:15:4:0:1:0:11:13:1a:0:12\
:f:17:16:0:0:73 < /dev/prolite
echo "<ID01><PU><FD>Server Up" > /dev/prolite
echo "<ID01><DTB>" > /dev/prolite
echo "<ID01><TA>00001U" > /dev/prolite
echo "<ID01><TD><FB><CC>Server Down" > \
   /dev/prolite
date "+<ID01><TB>*%H%MD" -date "7 min" > \
   /dev/prolite
```

To do this relies on some tricks. First, you cannot use a generic timer (**<TA>*****) for running standard messages because it conflicts with our watchdog timer. Secondly, we cannot use timer **<TA>00000** because it confuses the sign. Thus, we have to use one minute after midnight on Sunday (**<TA>00001**) in order to display our messages. When no timers are defined, the first timer defined shows our page.

Deleting timer B keeps us using timer A, which shows normal text. The sign ignores requests to delete timers and pages that don't exist. Once page A is defined and displaying, we define page B in the background and set up a time, again using the clever **date** command to output it seven minutes from now. Since our cron job is set to run this script every five minutes, the timer should never go off unless something is wrong.

If Linux suffers a power failure, the cron daemon is killed, or the sign becomes disconnected, it will display a warning message. The tag **<FD>** indicates the message should instantly appear instead of scrolling, where **<FB>** indicates the text should appear from the center—this will get your attention quickly.

More information about the protocol, cable and sign can be found at http://wls.wwco.com/prolite/. This site also includes source code for various applications that manipulate the sign.

Walt Stoneburner currently works as a software engineer for Downright Software, LLC. In his spare time, he enjoys working with Linux, playing non-computer games, reading and reviewing hardware and development software. Feel free to contact him at wls@wwco.com or ICQ# 5368391.

Archive Index Issue Table of Contents

Advanced search

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

PPR: PostScript Printer Spooling

Olivier Tharan

Issue #62, June 1999

Mr. Tharan tells us how to use the PPR spooler for large networks.

PPR is a PostScript printer spooler. It allows users to queue jobs for printing on different PostScript printers. A single host can be the printing spooler for a whole department or even a campus. It is well suited for managing a large number of printers.

Its main advantage over traditional printing servers, such as the System V LP or the Berkeley LPD, is that PPR is able to use every feature of the PostScript standard or the Adobe Document Structuring Conventions (DSC).

PPR Presentation

While many Linux users are pleased with the **lpd** printing daemon, I found it was not as easy to use when it came to support four TCP/IP HP 5M PostScript printers with more than 160 Linux, Macintosh and Windows computers on my LAN (local area network).

PPR has been designed specifically for the purpose of operating PostScript printers, accepting jobs from different systems such as UNIX (remote or local), Windows with Samba or Macintosh with AppleTalk. PPR can talk to any PostScript printer connected via a parallel port, a serial port, AppleTalk or TCP/IP. It can even talk to a remote lpd server. Beginning with version 1.00 (the first public release), PPR includes a GhostScript interface which enables it to print on every non-PostScript printer supported by GhostScript. PPR is therefore very well adapted to a heterogeneous environment.

PPR was designed with optimizing the use of PostScript and the Adobe DSC in mind. As such, PPR is able to stop a job if the printer indicates that the job contains a PostScript error (a queue listing will show “arrested” for that job). Many PostScript interpreters are “cool” about the errors one can find in

PostScript files, but a program generating PostScript should be strict. Remember that PostScript is a programming language. What if **gcc** let you forget a semicolon or undeclared variables?

Moreover, for printers that support it, PPR is able to capture the error messages sent by the printer and process them, sending a notice to the user or the administrator if something went wrong. After a job has completed, the user can be notified by various means: a mail message can be sent if he's a UNIX user; a WinPopup if he used Samba, and so on, so that batch and queue printing take their true meanings here.

If a user submits a non-PostScript file to PPR, it converts the file to PostScript using a set of tools determined at installation time. For example, a .tex file will be passed through LaTeX and **dvips** if these programs are available on the system; a .gif file could be passed though the **netpbm** utilities, etc. Then, PPR hands the converted file to the printer.

The queue handling is also well done. Any user can hold or cancel his job once submitted; any administrator (any trusted user belonging to a certain access list) can hold, release or cancel a job. Finally, a job can be moved from one printer to another one if desired.

PPR has also support for PPD files (PostScript Printer Description files), which are a convenient way to describe a printer's capability in just one file. PPR reads a PPD file and determines automatically what type of paper the printer can handle, what sort of bins it can use, and so on. PPD files can be downloaded from Adobe's FTP site, or you can have generally have them on the drivers' disk shipped with your printer.

Installation and Configuration

As of January 1999, the latest version of PPR is 1.40a4 and Linux is the primary development system. Version 1.31 and later are stable enough to be used in a production system. PPR uses very little resources other than the spool area, the size of which depends on how many users print on the site. We used to run PPR on a Pentium 66 Linux box which was also serving e-mail and Web. We have moved to a more powerful machine only because we felt it was better to separate the printing spool from the mail spool.

You can find the PPR FAQ at http:tarzan.trincoll.edu/printing/pprfaq.html and all the documentation at http:mouse.trincoll.edu/ppr/docs/index.html. They are written in an easy to understand style.

Installing PPR is easily done by following the steps described in “Installing and Using PPR” which is a beginner's guide to running PPR. Compiling and installing

the program in place is not difficult. If you plan on using AppleTalk with PPR, you will need AppleTalk support (either from NetaTalk or CAP, depending on the platform), and a separate library called NATALI (if you use Netatalk), shipped with PPR.

Next comes the configuration process. You may need to add a user and a group to your system files for PPR to work correctly. An access list system allows the administrator to delegate some powers; thus, he can assign some trusted users the task of cleaning up the printers' queues from time to time, without constantly requiring the administrator's help. There used to be some UNIX groups to which one had to belong in order to have PPR privileges, but this has been phased out in favour of the access list system.

Every printer on your network must be defined as well as the interface to use: **tcpip** for TCP/IP or **atalk** for AppleTalk. Then set some configuration variables, such as selecting a PPD file suitable for the printer or even adding a comment for the current printer.

You can even group printers and define groups. Jobs submitted to a group will go to the first non-idle printer in the group. The printers can even be rotated: jobs will be submitted in a round-robin fashion to each printer in a group.

Next, you can configure your system for different PPR access methods. With Samba, for example, the program **ppr2samba** is included which generates a simple file describing all the PPR-defined printers; include this file in your smb.conf file.

To add support for your AppleTalk users, you will have to launch the **papsrv** daemon which lists the printers on the AppleTalk network and handles jobs sent to PPR. You can even throw away your lpd server and use **lprsrv** to serve lpd jobs the same way papsrv does for AppleTalk.

PPR also allows for quota charging on a per-user basis, if you are concerned about who is doing what or if you simply want to charge for every sheet of paper coming out of your expensive printers.

Conclusion

PPR proves to be useful in a wide variety of configurations and should not be difficult to adapt to your needs. We use it here on a daily basis with very little maintenance: the main printing problems come from the printers themselves.

Many thanks to David Chappell, the author of PPR, who kindly reviewed this article.

Resources

Olivier Tharan (olive@minet.net) was a student in a French telecommunications high school and has been a system administrator for the students' campus network for three years. He enjoys getting e-mail, so he has subscribed to a dozen mailing lists which keep him occupied a good part of his short nights. Aside from that, he translates some Linux HOWTOs into French in his spare time.

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

Linux in Schools

Rob Bellville

Issue #62, June 1999

How a K-12 school system is using Linux to supply a myriad of stable network services to its students and staff.

The Millbury Public Schools make up a small school district of just over 1800 students, located in the heart of Massachusetts. We provide state-of-the-art computing services to our staff and students as a tool for learning and research. Linux holds a distinguished position on our network and supplies many of the network services required by our users. I hope this article will persuade educators to think of Linux as a viable and even preferred alternative to conventionally marketed products.

Like most public school districts, our computer technology was exclusively Macintosh-based and required a lot of effort and attention to keep it working and usable. After some lobbying on my part, an endeavor was made to bring our technology resources in line with what was in use in the “real world”. The argument I used was not that it was a nightmare supporting a hundred Macintoshes (it was), but that as an educational system we should be educating our students on computer systems they would be more likely to find in the college and business world. The fact that Apple Computer has been floundering in the marketplace didn't hurt my argument.

Coinciding with our migration to an Intel platform was the availability of a generous grant made by the Commonwealth of Massachusetts, specifically earmarked toward networking the state's school systems. This allowed us to set up a district-wide network and network services that rival those of many businesses. Between the state's grant and our supportive school committee, which approved our budget for new PCs, we had sufficient funds to do the job right.

The Details

Our network is arranged into 10Base-T workgroups tied into a central Ethernet switch (collapsed backbone configuration). This switch is segmented into two networks—administrative and student—providing our first level of network security. The servers are connected via 100Base-T to the switch. This arrangement allows us to share network resources without sharing bandwidth. The central switch creates a smaller collision domain for each workgroup and keeps traffic on the backbone to a minimum.

The client machines, which run Windows 95, are Pentium 166MHz clones with 32MB RAM and 2GB of disk space. We also have a number of Power Macintoshes remaining in some of our labs. Our three servers consist of Pentium Pro 200MHz with 128MB RAM and 4GB of SCSI disk space (soon to be 9+GB). One of these servers is a Windows NT machine used for applications serving and the other two are Red Hat Linux-driven for everything else.

The two Linux servers are set up to supply file and Internet services to our district. Our first box (http:www.millbury.k12.ma.us/) is set up as our WWW, FTP, e-mail and DNS server and resides on the DMZ of our Sonic Interpol firewall that is in turn connected to an ISDN router. This firewall/router combination would also have been Linux except for the fact that the Sonic Interpol offers content filtering. This helps us protect students from inappropriate web sites on the Internet—an essential device when many curious kids are around.

The second Linux box, located behind the firewall, was just recently “saved” from being an NT machine and operates as a proxy/cache server, intranet server, file server and has a number of other small duties. Luckily, NT was so disappointing in performance and rather unstable that it was “born again” as a Linux server. I have not regretted it.

Since both servers run Samba, I can map drives on the Win95 clients so that the transfer of files to our web site is a breeze. File sharing is also handled this way. Few users actually realize they are saving to a network drive, due to the speed of the transfer.

As we still have a few remaining Macintoshes on our network, we could supply AppleTalk services with Linux. We do not at this time, because these Macintoshes are located in the lower grade levels and our youngest students do not need most network services. It would be very easy to set up if it became necessary. Many of our network services are platform independent.

The Results

Linux running the Squid proxy server (http:squid.nlanr.net/) has increased the perceived bandwidth of our ISDN by a huge amount. It is also considerably faster than when it was running NT and Netscape's Proxy Server. Approximately 50-60% of all our web page requests are serviced by our in-house proxy server. The increase in speed resulting in pages coming through from the proxy at Ethernet speed rather than from the web site at Internet speed has simply delighted my users. I've found that this service has made the biggest difference on our network. I almost never hear a complaint about the Internet being slow.

Samba (http:samba.anu.edu.au/samba/) is a terrific program, and once you get used to setting up its configuration file, it is a charm to use. Setting up users in Linux is more desirable for me than it is with NT. That, coupled with the lack of additional costs for NT client licenses, makes Linux all the more attractive. This cost savings is enough to help pay for staff training in beginning UNIX system administration.

This brings me to another point. Nearly all schools operate under tight fiscal constraints. Linux offers a way to develop technology without dipping too deeply into the shallow school budget. Linux even recycles older hardware that can be obtained through donations from local companies.

A side effect of all this Linux use is a number of students have actually become interested in Linux and are setting up their own boxes at home. One of our technology specialists (a former Macintosh user) has also fallen in love with the command line after seeing me use the power of TELNET to add users, check my e-mail and reconfigure something on the box.

The Future

Now that I have firmly entrenched Linux into our network, I plan on further purges of Windows NT from our domain. We will be purchasing additional servers for our workgroups and offices next year, and where application servers are not required, Linux will be installed.

I also hope to recruit more students and introduce them to Linux. Enlisting a new generation of Linux enthusiasts will only help drive Linux more firmly into the future.

I am very grateful to the whole Linux community for their excellent software. A big thank you is also sent to the Samba team for their labor in helping me rid our computers of NT wherever possible. I would also like to applaud the Squid developers for a top-notch product.

I would like to hear from anyone who has ideas on how to make Linux even more attractive and useful in a school environment. I would also be happy to share my experiences with anyone thinking of implementing Linux in their school.

Help carry the Linux torch!

Rob Bellville is the Manager of Information Systems and Technology at the Millbury Public Schools, Millbury, MA. Having successfully escaped from the clutches of employment in the high-tech corporate world, he now enjoys infiltrating technology into the educational system and making school more fun, while getting most of the summers off! Rob can be reached via e-mail at rob@millbury.k12.ma.us.

Archive Index Issue Table of Contents

Advanced search

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

Linux in an ERP World

Uche Ogbuji

Issue #62, June 1999

Mr. Ogbuji takes a look at enterprise resource planning and Linux's place in this market.

Enterprise Resource Planning (ERP) is the latest hot computer topic for businesses. In the boardroom, it is often an even hotter topic than the Internet. ERP describes a class of computer systems that attempts to manage all aspects of an enterprise's operations. The basic structure of a typical ERP system is shown in Figure 1. Note that this diagram is highly generalized; actual ERP implementations differ wildly from vendor to vendor, and even from installation to installation. As you can see from the diagram, ERP systems cover an enormous scope, so they are very large software systems. They also tend to be relatively monolithic, which is their main selling point. This is also the feature that worries IS workers the most, and the feature that might make a Linux user scratch her head.

Figure 1. Typical ERP System

Frankly, ERP systems are typically sold with a pair of rose-colored glasses as an accessory. A gentleman I spoke to, who works at an enormous international beverage company, told me how the firm was sold an ERP system. The salesmen for a leading ERP vendor took the CEO to a demo room, with projection monitors showing simulated real-time charts of all of the company's operations and finances. The salesmen told the executives that with this product, they could have their fingertips on all of their company's data. The sale was closed at that point. Now, years later and a scandalous sum over budget, the company has little choice but to complete the installation. The IS staff is completely disillusioned, but apparently not the executives, who are still waiting for the magic cockpit to go on-line. This story is all too typical. ERP systems are sold mostly at the executive level, and skeptical IS staff have little to say other than the narrowest, technical implementation details. Furthermore, the process of putting together an ERP system is such an enormous undertaking and

commitment that once the project is begun, there is no backing out without disastrous financial and operational results.

Where does Linux fit into this rather alien world? Many Linux professionals will find themselves caught up in an ERP installation at some point in their career, since ERP systems are selling very well these days. But Linux makes the hapless IS professional think of a more fundamental issue: what brought about the whole ERP trend?

If you look at the history of computer software systems, you will notice that the industry has not treated business very well. Competition among traditional software vendors often involves wars to promote one data format or process over another, with the reasoning that the success of proprietary data formats accelerates the market share of proprietary applications. Many departments bought computer systems to serve their particular operational needs. The accounting department got machines and software that run the most effective accounting software. The technical support department got systems that manage incoming calls and problem resolution. The marketing department often got Macintoshes for the graphics and layout software.

These individual packages served each department well enough, but it's a small leap of business sense to deduce that the real gains in productivity would come if the various departments could exchange data. Suppose quality assurance could recognize a defect trend from the technical-support data and tighten testing, or that the accounting department could verify purchase orders for raw materials directly from manufacturing resource-planning systems which would determine how much of a product to manufacture from marketing's sales projections. However, when CEOs tried to implement such interactions, they quickly ran into barriers of differing data formats, conflicting business rules and physically disconnected systems. When they went to the vendors of departmental systems for help, they typically didn't get far. Not only would it be prohibitively expensive for each vendor to integrate with every other system out there, but vendors imagined a competitive advantage to not doing so.

The furor over software standards is a recent phenomenon. Historically, software standards have been very slow to emerge, motivated more by temporary alliances for market share than by any real desire for interoperability. The typical solution for interoperability was to buy entirely from one vendor. Manufacturing companies have very specialized needs for data exchange, so about a decade ago, a few smart companies started offering manufacturers a suite of software that handled all aspects of their operations with the assurance that information would be compatible between departments. This was the birth of ERP. Since then, pioneers such as SAP and Baan have grown prodigiously. From roots in manufacturing, they now sell to

almost every major industry. Other companies, sensing profit, have joined the ranks of ERP vendors: PeopleSoft from the human resource world, Oracle from the RDBMS world and JD Edwards from the financial systems and business services world.

Now, most Linux users might see all of this as a huge over-correction: organizations can't get departmental computers to collaborate, so they scrap them all in favor of supersystems from single vendors that claim to do all of the organization's data processing. Linux users are accustomed to standards and a culture of software collaboration, but these trends are only now slowly beginning to permeate the rest of the software industry.

The entire ERP trend does pose some tough questions for Linux advocates. First of all, Linux's infiltration into enterprise has often come at the departmental level, usually on departmental web servers. As more companies adopt ERP systems and insist that internal and public web content feed into the same enormous data stream, Linux may be harder to push in the workplace. Also, for Linux to move from the web server to the application server environment, it will often need to go head-to-head with an ERP module—a battle for which Linux is politically ill-equipped.

Linux is just an operating system, and being UNIX-compatible should theoretically not be far from running ERP systems, which usually have commercial UNIX ports. Many hints and whispers have been reported on Slashdot.org and elsewhere that various ERP vendors have development versions of their server modules running on Linux. This is to be expected and follows from the interest of so many DBMS vendors in Linux, but again, it doesn't address an issue that is of as much importance to the Linux community as the running platform. The culture of Linux, promoting distributed systems, narrow scope for applications and standards-driven collaboration between applications is under stealthy attack by the ERP approach. This is a very important issue for Linux and open-systems advocates to address, because ERP vendors are almost as wealthy and resourceful as Microsoft and just as anxious to press their proprietary systems. It is in the interest of ERP vendors to continue making it attractive for customers to buy all of their major information systems from them. On the other hand, Linux users typically insist on buying or downloading various software packages for various purposes and expecting them to work well together. This conflict is most apparent in small-to-medium businesses, which are the latest target of ERP vendors, but probably the most likely to gain from the open and distributed systems approach. This so-called “mid-market” is comprised of companies with $250 million to $500 million (US) in annual revenues.

ERP vendors are pushing the myth that an enterprise needs a massive system running on a tight cluster of machines for effective computing. The Internet is but one recent development that proves distributed computing can be just as productive as highly centralized computing. Departmental systems are increasingly feasible in the age of Intranets, driven by such interoperability standards as XML, CORBA and LDAP. Databases are now almost all reliably SQL, a more recent phenomenon than one might imagine, and many domain areas are developing forms and process standards for Electronic Data Interchange (EDI) and other initiatives.

Make no mistake about it; Linux does have some definite short-comings for enterprise computing. Some are only in perception, such as the 2GB file-size limit. This, of course, applies only to 32-bit systems, such as Intel x86 platforms. Major database vendors already know how to work around this limitation, but Linux on Intel gets most of the press and many people in a decision making capacity read pundit columns that complain of the supposed limitation. Even if one sets miseducation aside, there also need to be more fundamental processes, standards and drivers at the system and server-device level for mirroring, redundancy and fail-over because an enterprise system must be bullet proof, even by Linux's high standards. Intel's recent attention to Linux provides a much-needed boost towards addressing such high-availability issues.

DBMS vendors are also an important part of this process. Enterprise computing, whether using single-source ERP or distributed information systems, is all about data management. DBMS vendors can take leadership in tweaking Linux into an “enterprise-class” operating system. Their researchers can provide kernel patches, and they have the means and need to test enormous transaction loads. As a true open-source advocate, I take a moment to mention PostgreSQL, which is an open-source database. It still doesn't support some features typical in enterprise-class systems, but its feature set is tremendously robust, and a good argument against detractors who like to consider open-source efforts as good only for hobbyists.

It is not the specific applications that Linux is lacking. We don't need a group to get together and write an open-source resource planning application. What Linux needs is the basic underpinnings: the infrastructure for such applications. Its current momentum will surely envelop many vendors of vertical applications, and such applications better served by the open-source model will naturally tend in that direction. For instance, Linux needs attention from developers of middleware and transaction-processing systems. These systems have been standardized enough that they could reasonably support open-source projects, and they are an integral part of the ERP equation. Another area of importance is business rules.

One reason ERP installations are so precarious is they usually involve business-process reengineering. ERP systems are complex enough as they are. If they had to encompass all the company rules, policies and customs of every business out there, their complexity would be practically infinite. Instead, each ERP vendor has picked its own set of “best practices”, business rules which it believes to be most effective and has enshrined these in its systems. If a company wants to adopt an ERP system, it must reform its own business processes to those of the ERP system. This usually involves a major effort in paper shuffling, training and testing. Many CEOs, who feel their companies have chaotic or non-existent business logic, seek to implement ERP just for the reengineering process. In fact, this is often painful and resource-intensive.

Figure 2. Linux ERP System

If a company has good operational practices, it does not need to go through the pain of remodeling itself in some other's image. Very often, a company's unique approach to business is part of its competitive edge. This is a strong argument for decentralized computing. If Linux is to use this argument, it will have to provide a powerful system for managing business rules and incorporating them into enterprise systems, as described in Figure 2. Note that once broken down to the departmental level, Linux has the capability and competitiveness to host all of the functions displayed. What is needed is the applications to fill in the blanks. The enterprise-class databases are coming, but the data-exchange management and executive information systems are a tougher proposition. For vendors to prepare such systems for Linux, a solid framework for business rules would probably have to be in place. This may sound easy enough to implement, but in its purest form, it is one of the holy grails of computer science. Research efforts ranging from graph theory to artificial intelligence have sought good ways to enshrine arbitrary business rules (or even just common-sense rules) into a computing framework. A system that could handle enterprise-class computing would need the ability to analyze business objects and apply various operations and constraints on them, as well as check pre-conditions, post-conditions and invariants. This is an area where the great problem-solving ability displayed by distributed developers such as Linux's can make strides.

There is no doubt, however, that those of us in the Linux community face tremendous financial interests in an entrenched industry if we decide to compete. ERP is a very profitable business, in contrast to the plummeting margins in shrink-wrap software. ERP systems are pushed by the companies that write the software, consulting companies (including the consulting arms of the big six accounting firms) and hardware/systems interests such as Hewlett-Packard and IBM (see “An Unlikely Ally”). These same companies provide reengineering and customization services. The frenzied marketplace ensures

that consultants can often command $200 to $300 US per hour. The total cost for ERP installations ranges from $100,000 for the smallest organizations with the least reengineering effort required (often called “accelerated” implementation) to as high as $500 million for large multinational firms. Note that I mention customization with regard to the implementation effort. ERP installations often demand as much effort in customization as is required to write the information systems from scratch using off-the shelf DBMS, middleware and design tools. ERP vendors encourage customers to customize as little as possible, so as not to break everything when the APIs change, but the reality is that every business has its quirks which necessitate customization.

Much of the recent effort in the Linux world has gone toward making Linux a feasible desktop replacement. To observers of modern enterprise-wide computing, this seems almost like misplaced energy. Intranets and other such developments are slowly turning front ends into commodities, and the real power is shifting from the client back to the server, although the server does not have to be as monolithic as a mainframe. It is important for Linux developers to turn their attention to the infrastructure which is increasingly being demanded by enterprise-class computing and the executives who demand it.

Many will argue it is not important for Linux to prove itself in this market, but if you believe in the superiority of collaborative computing, you should recognize that this ideal has the potential to be snuffed out by sheer misguided politics and economics. If we want that server in the corner of the data center to ever run anything besides Apache and Samba, we might want to channel some of the boundless Linux energy to the enterprise.

Linux users face an interesting dilemma. A sector of the market that has traditionally neither interested nor challenged us is beginning to shape trends that could affect the future of computing, as much from a cultural as from a technological view. To ensure the encouraging growth of Linux and the associated improvements in business attitudes toward information technology, developers would best address some of the needs of enterprise computing. Commercial developers will certainly fill in the gaps. Then perhaps the promise offered by the client/server movement, more flexible and decentralized computing, will stand in a little less danger from huge, single-source ERP systems.

An Unlikely Ally

Uche Ogbuji is co-founder of FourThought LLC (http://www.fourthought.com/), a consulting firm specializing in custom software development for enterprise applications, particularly Web-based integration platforms for small or medium-sized businesses. He can be reached via e-mail at uche@fourthought.com.

Archive Index Issue Table of Contents

Advanced search

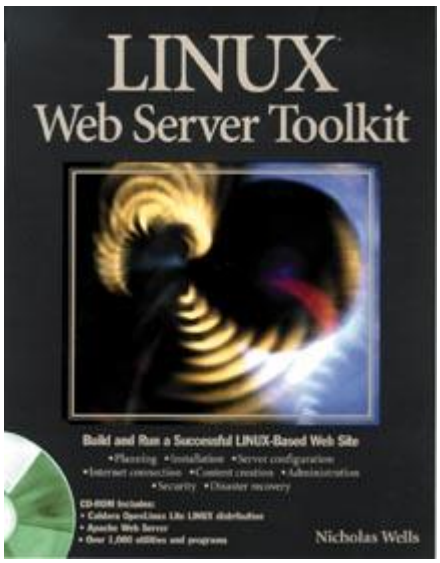
Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Book Review: LINUX Web Server Toolkit

Keith P. Solla

Issue #62, June 1999

A review of the *LINUX Web Server Toolkit*, a book that takes the reader completely through the procedure of building a web server.



- Author: Nicholas Wells
- Publishers: IDG Books Worldwide, Inc.
- URL: http://www.idgbooks.com/
- Price: $39.99 US
- ISBN: 0-7645-3167-0
- Reviewer: Keith P. de Solla

The *LINUX Web Server Toolkit* takes the reader completely through the procedure of building a web server, from planning to disaster recovery. The book comes with a CD containing Linux, Apache and a multitude of scripts and programs. The copy I received came with Caldera OpenLinux Lite, but I have seen bookstore copies with Red Hat Linux. This book provides a reasonable overview of all issues involved with setting up a web server. It cannot, of course,

cover topics in the same level of detail as books such as O'Reilly's *Apache*, but it was not meant to do so.

The author assumes the reader has access to a computer, but does not assume detailed knowledge of Linux, web servers or the Internet. A novice should be able to use the information in this book to set up a web server, and an experienced user will also find information of value.

The 21 chapters are grouped into four parts: planning, installation, maintenance and adding advanced features to your web server.

Part I: Planning Your Web Site

Chapters 1 and 2 discuss non-technical issues related to creating a web site for a business and provide some background technical information for novices. Chapter 3 sells Linux as the choice platform for a web server, followed by a discussion of UNIX and Internet terminology in Chapter 4. Part I can be skimmed (or skipped) by more advanced users.

Part II: Installing Your Web Server

Chapter 5 reviews hardware requirements, then takes the reader through the steps of installing Linux. As I had no need to install an older version of Linux, I did not install anything from the CD. The book is not a replacement for a detailed installation guide, but does provide sufficient information so that a novice can install the software. While sparse (nine pages), the section on configuring XFree86 is quite well-written and should get most people through the “fun” of creating an XF86Config file. However, it does not provide a troubleshooting guide, so users with non-standard or “problem” hardware will need to read the HOWTOs and Release Notes. One important item missing is a warning that probing the video hardware may hang the computer. Novices need to know this, especially after learning that driving the monitor at too high a frequency can damage it. A brief discussion on network configuration follows.

Chapter 6 takes the user through installation and setup of both Apache and Netscape FastTrack web servers. FastTrack is a commercial product not included on the CD. Finally, Chapter 7 builds on the basic terminology of Chapter 3 and looks at connecting to the Internet in more detail. Over 90 pages cover everything from “what being connected means” to setting up DNS. The emphasis is on the requirements and available options for setting up a commercial web server.

Part III: Maintaining Your Site from Day to Day

Business issues such as advertising, search engines and gathering data are discussed briefly in Chapter 8; readers setting up personal web sites can skip it. Basic HTML is covered in Chapter 9, but not in any great depth. URLs are provided for HTML authoring tools and other information, but enough detail exists to allow a novice to create basic pages with tables, links and graphics. This information is expanded in Chapter 10 to include web site scripts and forms.

The rest of Part III (Chapters 11 to 14) covers configuration of the web server, additional services and collecting statistics. Chapter 11 is specific to Apache and provides a thorough overview, but the material is rather dated. (Apache v1.1.1 is included on the CD.) If you intend to set up a web server, I advise purchasing the O'Reilly book on Apache and visiting these web sites: http://www.apache.org/ and http:/www.apacheweek.com/.

As most people seem to be using Apache, I skipped Chapter 12 which describes configuring Netscape's Fastrack server. Chapter 13 discusses the pros and cons of web site statistics and provides URLs for web server statistical tools. Finally, Chapter 14 covers additional services such as FTP, e-mail, gopher and WAIS.

Part IV: Adding Advanced Features

A very brief (eight page) introduction to Java and JavaScript is given in Chapter 15. Version 1.0.2 of the JDK is provided on the CD, but a list of related web sites would have been helpful as well. Those interested in Java and/or JavaScript will want to look for books on those specific topics. Almost as brief, but more detailed, Chapter 16 discusses gateways (such as e-mail and database) and provides lists of sites for gateway software. The novice will be able to learn enough about gateways to understand what they do and whether one is required. The next chapter gives a quick overview of application programming interfaces (API) for Fastrack and Apache. APIs allow the user to extend the capabilities of the server. More detailed documentation will be required by those wishing to actually do this.

Chapter 18 concerns the all-important issue of security. It begins with a summary of types of attacks, both generic and web-specific. A checklist of tests to try and files to check gives the novice a good starting point for reviewing site security. This is followed by a brief discussion on firewalls. The reader is then pointed to an on-line firewall FAQ and http://www.yahoo.com to search for more information.

The remaining chapters deal with issues of web maintenance, backup and Linux package upgrading. Chapter 19 includes a list of HTML validation tools

and recommends HTML Analyzer for automated checking of your web site files. The book finishes with a description of the CD-ROM files in Appendix A.

CD-ROM

The CD included with my copy of the book contained complete, but somewhat dated, software. For example, it installs kernel v2.0.29, Apache v1.1.1 and v1.0.2 of the Java Development Kit. However, this book is hardly unique in this respect—users will generally buy or download the latest releases elsewhere. The important issue is the CD provides all the software necessary to install and set up an Apache server on a Linux 2.0.*x* kernel. Some additional tools are included on the CD including (much to my surprise) Xemacs. I would like to see Xemacs included on more CD sets.

Conclusion

The book provides a reasonable overview of the issues and mechanics relating to implementing a web server. The target audience is beginner to intermediate-level users. If you are computer literate but a web novice, this book contains sufficient detail to enable you to set up a web server. The depth is such that more advanced people will also find the book useful, but it will not make someone an expert on Apache or Java. Throughout the book, URLs are provided so the reader can obtain more information, documentation or software related to the specific topics being discussed. This is especially useful given how quickly a printed book can become dated. If you're interested in what is involved in setting up a commercial web site, this book is a very good place to start.

Keith P. de Solla, P.Eng is an underemployed VLSI CAD Engineer, currently masquerading as a Linux guy. When not doing computer stuff, he can be found engaged in the politically incorrect (but really fun) activity of action pistol shooting. He can be reached via e-mail at kdesolla@cyberus.ca.

Archive Index Issue Table of Contents

Advanced search

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