;login:

THE MAGAZINE OF USENIX & SAGE

July 2001 • Volume 26 • Number 4

UNIX users meeting unites NBI, ISI engineers

inside:

USENIX & SAGE

The Advanced Computing Systems Association & The System Administrators Guild

USENIX Upcoming Events

10th USENIX Security Symposium

AUGUST 13–16, 2001
Washington, D.C., USA

http://www.usenix.org/events/sec01/

Registration materials now available

5th Annual Linux Showcase and Conference

Co-sponsored by USENIX & Atlanta Linux Showcase, in cooperation with Linux International

NOVEMBER 6–10, 2001
Oakland, California, USA

http://www.linuxshowcase.org

Registration materials available: August, 2001
Final Papers due: September 14, 2001

XFree86 Technical Conference

(Co-located with the 5th Annual Linux Showcase & Conference)

NOVEMBER 7-8, 2001
Oakland, California, USA

http://www.usenix.org/events/xfree86/
Registration materials available: August, 2001
Camera-ready final papers due: September 14, 2001

15th Systems Administration Conference (LISA 2001)

Sponsored by USENIX & SAGE

DECEMBER 2–7, 2001
San Diego, California, USA

http://www.usenix.org/events/lisa2001

Registration materials available: September, 2001
Camera-ready final papers due: October 1, 2001

First Conference on File and Storage Technologies (FAST)

Co-sponsored by USENIX, IEEE TCOS, and ACM SIGOPS

JANUARY 28–29, 2002
Monterey, California, USA

http://www.usenix.org/events/fast/

Notification to authors: September 14, 2001
Registration materials available: October, 2001
Camera-ready final papers due: November 15, 2001

BSDCon 2002

FEBRUARY 11-14, 2002
San Francisco, California, USA

http://www.usenix.org/events/bsdcon02/

Abstracts due: August 27, 2001
Notification to authors: October 1, 2001
Camera-ready final papers due: December 4, 2001

The Fourth NordU/USENIX Conference (NordU/USENIX 2002)

Sponsored by EurOpen.SE and USENIX

FEBRUARY 18-22, 2002
Helsinki, Finland

http://www.nordu.org/NordU2002

Extended abstracts due: September 7, 2001
Notification of acceptance: October 12, 2001
Final papers due: December 7, 2001

The Systems and Network Administration Conference (SNAC)

Sponsored by USENIX & SAGE

MARCH 17-21, 2002
Dallas, Texas, USA

2002 USENIX Annual Technical Conference

JUNE 9–14, 2002
Monterey, California, USA

For a complete list of future USENIX events, see **http://www.usenix.org/events**

contents

;login: Vol. 26 #4, July 2001

;login: is the official magazine of the USENIX Association and SAGE.

;login: (ISSN 1044-6397) is published bimonthly, plus July and November, by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $60 per year.

Periodicals postage paid at Berkeley, CA, and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

©2001 USENIX Association. USENIX is a registered trademark of the USENIX Association. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this publication, and USENIX is aware of a trademark claim, the designations have been printed in caps or initial caps.

Cover photo: Early FRUUG Meeting
(see page 73)

motd

Stirring the Pot

by Rob Kolstad

Dr. Rob Kolstad has long served as editor of *;login:*. He is also head coach of the USENIX-sponsored USA Computing Olympiad.

<kolstad@usenix.org>

I try hard to share positive messages. People accuse me of being "unrealistic" or even call me "Pollyanna" for my occasional assertion that "each day is better in each and every way." I guess I'm temporarily abandoning that for this column. I fear I will stir the pot a bit.

The world has so many targets to shoot at. Let's start with the proliferation of licenses for "open source" and other sorts of software.

The GPL would have people who benefit from software return back to the development community any and all modifications they make. Doesn't that sound nice? Freedom to use software, financially free updates from those who repair it, what could be better?

You probably know that I was president of BSDI (a company marketing a Berkeley-style OS). BSDI's software division has been acquired by Wind River Systems. I haven't participated in BSDI strategy meetings since my departure a couple years ago, so I think I'm free to speculate as to what's going on.

Why didn't Wind River choose Linux for their UNIX story? Zero dollar licensing is complemented by plenty of support from the development community. What possible objection could Wind River have?

I imagine they objected to the rather stringent licensing terms of the GPL. I believe that they, as a company approaching half a billion dollars per year of revenue, felt that they should gain some return on the R&D investment that they would be putting into their operating system in order to turn it into a commercially viable, documented, tested – maybe even certified – product that has not only pre-sales support and actual marketing and sales behind it but also a post-sales support staff that is dedicated to helping customers get their commercial applications running. I have to believe that Wind River felt that giving their R&D – their intellectual property – out to the world for free (of course, they get others' fixes and enhancements back, too) just wasn't a winning financial proposition.

I sympathize with this position. I tried to run a small company selling BSD-style software and support. It's an extremely challenging game! Well, at least for me it was.

So, I believe the GPL supporters need to believe that they're only going to see GPLed software in systems that have specialized hardware that is difficult to reproduce or have some other very high barrier to entry. Commercial (i.e., for-profit) companies have a hard time defending the public divulsion of their intellectual property (IP). Without IP, they face uphill battles every time funding is required. It's hard to run a company without funding; I'm an expert on that.

A Slashdot article mentions that a court test of the GPL is soon to come. It seems someone has taken some GPLed software from the net and built a product – but they refuse to part with the enhancements. That's dirty pool: the license is clear about the required tradeoffs. If they used GPLed code, they need to abide by the licensing terms! I hope the courts find that such clearly stated yet implicit contracts are valid. I don't want to live in the world where other, more complex requirements might be put on this sort of software development and distribution.

So what do we have? We have a huge set of publicly available software. We have more than a dozen different licenses, each with its own little flavor of freedom or other buzzwords. It's a fascinating world we live in, and the fascination only increases as people study and debate the merits of each one.

;login:

EDITORIAL STAFF

EDITORS:
Tina Darmohray <*tmd@usenix.org*>
Rob Kolstad <*kolstad@usenix.org*>

STANDARDS REPORT EDITOR:
David Blackwood <*dave@usenix.org*>

MANAGING EDITOR:
Alain Hénon <*ah@usenix.org*>

COPY EDITOR:
Steve Gilmartin

TYPESETTER:
Festina Lente

PROOFREADER:
Lesley Kay

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: +1 510 528 8649
FAX: +1 510 548 5738
Email: <*office@usenix.org*>
<*login@usenix.org*>
<*conference@usenix.org*>
WWW: <*http://www.usenix.org*>

apropos

Call It Like It Is

by Tina Darmohray

Tina Darmohray, co-editor of *;login:*, is a computer security and networking consultant. She was a founding member of SAGE.

<tmd@usenix.org>

One of the things you learn to live with as a graduate of UC Berkeley are jokes about political activism and far-left, "out there" behavior. I didn't choose Berkeley for that reason, but my friends and family do note a buck-the-system streak in me. Recently, when my kids brought home a standardized test form, I guess that streak took over. On the section that asked about ethnicity, I bucked the system and created my own category. The choices were the familiar ones: Asian, African American, Native American, Hispanic, White (not of Hispanic descent), and so on. These choices bother me because they are not consistent. For example, I figure that if I'm "White" then the other choices need to be things like "Yellow, Red, Black, Beige" and so on. Or if someone else is "African American" or "Native American" then the rest of them should go something like "European American, Asian American" and so on. Or you can do the "oids": Caucasoid, Mongoloid, Negroid, etc. But you gotta be consistent, or it sends an additional message (a topic for another magazine, I'm sure). So I crossed out "White" and wrote "European American" then sent the form in to the unsuspecting teacher. She probably fixed it for me.

Accurate naming conventions are important. Agreed upon nomenclature goes at least half the distance in facilitating fruitful problem-solving discussions. When you don't share a common "language" for a topic, it's very hard to get to the root of the problem and devise a solution-oriented plan of action. That was the problem I faced in 199X as a hiring manager at Lawrence Livermore National Laboratory. LLNL did not have a system administrator job description. As such, whenever I hired a new system administrator, I had to figure out which non-fit description to place them in. Existing descriptions in our general field, were operator, computer programmer, and computer associate. In over-simplified terms these were folks who hung tapes, cut code, or provided administrative assistance by using computerized applications for a living. None of them were a good fit for a system administrator.

The problem with the lack of an appropriate job description for the system administrators manifested itself in many different ways. The most troublesome for me was at salary and review time. During the review meetings, system administrators would be compared with their "peers" in whatever category they'd been placed in. And, no matter what category that was, they were under-performing. For instance a system administrator may have performed some backups over the course of the year, but not nearly as many as the operators had. Similarly, they may have written some shell scripts to automate system administration, but that hardly compared the with the computer scientists, who created and maintained huge libraries of code. And so their performance in their category was poor, the review was subpar, and the resulting salary increase wasn't much. Over time. this snowballed into all kinds of personnel problems such as low esteem, low retention, and difficulty in hiring. I wanted a job description that used the right title, described the right qualifications, and measured the right skills; I wanted the right nomenclature for our profession!

It was natural that I became interested in SAGE and the SAGE Job Descriptions. I felt they were the ticket I was looking for to address my own hiring and assessment woes at LLNL. Note that the players wouldn't necessarily change, just the agreed-upon naming and description of them. Over the years I've heard from many HR groups that this (re)classification of their system administrators was very helpful. This underscores my belief that baseline nomenclature is key. We're not alone in this, as taxonomy is the underpinning of many scientific and technical efforts. It's clear that common language is basic to common goals.

We're currently considering updating the SAGE Job Descriptions to make them less

UNIX-centric – turns out making them more neutral is the easy part. Expanding them to embrace other prevalent OSes might be more difficult. The crux of the problem is that other OS system administrators have different titles for "our" skill-set categories and some of our titles mean entirely different things to them. Probably the best example of this is "Network Administrator," which seems to mean two quite different things to UNIX and NT/Netware system administrators. A fundamental question facing us in this update is, "Do we fit their jobs into our categories, or do we maintain separate categories?" There are good arguments for each approach. Part of me feels that describing entirely separate categories increases the divide which we're actually trying to bridge.

I welcome your thoughts on this topic.

conference reports

This issue's reports are on the 6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS '01)

OUR THANKS TO THE SUMMARIZER:

Nanbor Wang

6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS '01)

San Antonio, Texas

January 29–February 2, 2001

Summarized by Nanbor Wang

Best Student Paper Awards

Yi-Min Wang of Microsoft Research, Co-Chair of the Conference, announced the best student award. Many of the accepted papers were written by students this year. Of them, two received identical scores, and both of them were awarded the best student paper award. They are:

"Content-Based Publish/Subscribe with Structural Reflection" by Patrick Th. Eugster and Rachid Guerraoui of the Swiss Federal Institute of Technology, Switzerland and

"Multi-Dispatch in the Java Virtual Machine: Design and Implementation" by Christopher Dutchyn, Paul Lu, Duane Szafron, and Steve Bromling of University of Alberta, Canada; and Wade Holst of the University of Western Ontario, Canada

Keynote Address

The Brave New World of Pervasive, Invisible Computing

Stu Feldman, Institute for Advanced Commerce, IBM T.J. Watson Research Center

In this talk, Feldman pointed out that, with their increasing popularity, portable computing devices have become pervasive in our daily lives. Nevertheless, unlike desktop computers, there is still no single computing model that dominates the mobile world nor should there be one since mobile devices have very different hardware interface designs. With the current trend, the number of mobile devices accessing the Web will shortly exceed that of desktops. There are plenty of opportunities to develop mobile devices, and Object-oriented (OO) technologies and middleware will be required to integrate pieces of software.

Feldman went on to point out that quality of service and immediacy will be important for next-generation mobile applications providing networked services, such as market monitoring, financial transactions, and real-time information broadcasting. The heterogeneity of the environment these applications must run on presents many challenges to developing these applications. The diversities in user demographics, device-specific user interfaces and business models processes of service providers further complicate the issue.

In conclusion, Feldman envisions a world of pervasive devices that are helpful, always on, always available, personal, friendly to use, easily authored, managed, secure, and reliable. It's up to the application and middleware developers to meet these challenges.

Session: Distributed Objects

TORBA: Trading Contracts for CORBA

Raphaël Marvie, Philippe Merle, Jean-Marc Geib, and Sylvain Leblanc, Laboratoire d'Informatique Fondamentale de Lille, France

Raphaël Marvie presented their work on the Trader-Oriented Request Broker Architecture (TORBA), which is a framework and a collection of tools to automate the use of the CORBA Trading Service. Marvie pointed out that locating a service is the essential part of distributed applications. Although the Object Management Group (OMG) already defined the CosTrading Service, it is complicated to use and does not support type safety. The proposed solution is to make the Trading Service an integral part of objects and provide tools to integrate the Trading Service into applications automatically.

TORBA defines the concept of *trading contracts* in TORBA Definition Language

(TDL). With the help of a TDL compiler, TORBA generates the code for *trading proxies*, which hide the complexity of using the OMG Trading Service and provide the necessary type-checking mechanisms. TDL allows application developers to specify the properties of an object and how the object will be offered to and queried for by its clients. Although TORBA introduces some extra operations when connecting and using the trading server, a local library is used to optimize its performance, and other ORB-specific optimizations can be used to improve the performance further.

For more information, contact *marvie@lifl.fr*.

Dynamic Resource Management and Automatic Configuration of Distributed Component Systems

Fabio Kon, University of São Paulo, Brazil; Tomonori Yamane, Mitsubishi Electric Corp.; Christopher K. Hess, Roy H. Campbell, and M. Dennis Mickunas, University of Illinois at Urbana-Champaign

Component technology is gaining popularity in today's computation environments. Software components increasingly collaborate with each other, and how they are configured often depends on the hardware that they run on. Managing software components is becoming complicated as dependencies among components and the number of hardware platforms they can run on grows.

Christopher Hess presented their work on a component configuration framework that provides mechanisms to help:

- Automatic configuration of component-based applications
- Intelligent, dynamic placement of applications in distributed systems
- Dynamic resource management for distributed heterogeneous environments
- Component-code distribution using either push or pull models
- Safe dynamic reconfiguration of distributed component systems

Hess also provided a performance analysis of their component configurator implementation.

For more information, contact *ckhess@cs.uiuc.edu*.

An Adaptive Data Object Service for Pervasive Computing Environments

Christopher K. Hess, Roy H. Campbell, and M. Dennis Mickunas, University of Illinois at Urbana-Champaign; Francisco Ballesteros, Rey Juan Carlos, University of Madrid

Christopher Hess continued to describe their work on the adaptive data object service (DOS). He first pointed out that remote data access is the most essential part in modern computing environments. The traditional file sharing mechanisms, however, are not suitable for many mobile devices and their computing model because they lack the necessary resources, such as bandwidth, memory, software, and CPU power to decipher the raw data stored in files. To address the problem, Hess's team proposes the adaptive DOS, which is inspired by the model-view-controller (MVC) model.

With DOS, data are represented and accessed through *containers* and *iterators*. Containers shield the actual file representations from the application. Data can either be remote or local. Containers can also be instantiated with *filters*, e.g., a GrepContainer, to change the user's view of a data stream. Iterators provide different ways to traverse the data in containers.

DOS is currently implemented as CORBA service and can be linked-in dynamically. XML descriptions are used to describe the capability of DOS objects, so a device knows what a DOS object is and how to use it. DOS is part of the Gaia system, which is an infrastructure that exports and coordinates the resources contained in a physical space and defines generic computational environments for ubiquitous computing devices.

For more information, contact *ckhess@cs.uiuc.edu*.

Session: Infrastructure

HBench:JGC – An Application-Specific Benchmark Suite for Evaluating JVM Garbage Collector Performance

Xiaolan Zhang and Margo Seltzer, Harvard University

In this paper, Xiaolan Zhang pointed out that as Java gains popularity several Java Virtual Machine (JVM) aspects, such as dynamic memory management and garbage collection (GC), can become problematic for performance sensitive applications. Understanding GC performance characteristics in order to select the right GC implementation can significantly affect overall application performance. Traditional GC benchmarking approaches may not measure the behavior of targeted applications accurately because they usually measure and compare the total execution times of a fixed set of applications using different GCs. These applications, however, may not have the same behavior of the application we are interested in.

Zhang presented their work on HBench:JGC, which is a vector-based benchmarking suite that characterizes application memory-usage patterns and the GC implementation independently. It then combines both metrics to evaluate the performance of a GC in the benchmarked application. The experiments show that HBench:JGC can predict the actual GC time within 10% for most applications.

Distributed Garbage Collection for Wide Area Replicated Memory

Alfonso Sánchez, Luis Veiga, and Paulo Ferreira, INESC/IST, Portugal

Alfonso Sánchez presented their work on distributed garbage collection (DGC) for

wide area replicated memory (WARM). He first identified the deficiencies of two traditional DGC schemes. DGC that uses the message-passing model fails to consider the existence of replicated objects. Conversely, DGC that uses the shared-memory model does not scale well, since it depends on sending causal information using the underlying communication channels.

The authors propose a DGC algorithm that adapts the classical reference counting and improves it to take into consideration replications.

For more information, contact *alfonso.sanchez@inesc.pt.*

Multi-Dispatch in the Java Virtual Machine: Design and Implementation

Christopher Dutchyn, Paul Lu, Duane Szafron, and Steve Bromling, University of Alberta, Canada; Wade Holst, University of Western Ontario, Canada

Christopher Dutchyn presented this paper on extending the JVM to support multi-dispatch. Mainstream OO languages like C++ and Java only support uni-dispatch, such as function overloading and virtual functions. Under these programming environments, programmers often must implement double-dispatch—which inspects the type or arguments of the event object—to decide how an event should be dispatched to the proper event handler. Supporting multi-dispatch greatly simplifies the structure of OO application programs.

Dutchyn's work extends the JVM support for multi-dispatch without changing the language definition. His approach also (1) maintains backward compatibility to source code and library and (2) isolates the semantic changes and performance penalty incurred by the extension within the places where it is used. Several other multi-dispatch designs are compared and contrasted with their design. Empirical results show that supporting multi-dispatch through JVM extension has lower latency compared to functionally equivalent handcrafted code.

For more information, contact *dutchyn@cs.ualberta.ca.*

Using Accessory Functions to Generalize Dynamic Dispatch in Single-Dispatch Object-Oriented Languages

David Wonnacott, Haverford College

David Wonnacott presented another paper discussing the applicability of dynamic dispatching using C++ as the programming language. He explained that the single-dispatching strategy used by most OO languages unnecessarily restricts the extensibility of programs. Specifically, the dynamic-dispatch mechanism (virtual functions) that C++ supports inhibits code reuse through inheritance. The Accessory Functions proposed in the paper allow dynamic-dispatching a message to a group of functions with similar signature but one *virtual* argument. This virtual argument is used to determine which function in the group to dynamically dispatch the invocation to, whereas current OO languages use the receiver of the message to determine how the message should be dispatched.

Accessory Functions work with a group of classes without requiring the implementation details of those classes or violating their encapsulation. The cost of dispatching to accessory functions should be no more expensive than existing language constructs for dynamic-dispatch. Like C++, the system must be able to produce errors related to dispatching prior to program execution. The dispatcher extension should dispatch the message based on the dynamic-dispatch semantics existing in the language.

The alternative dynamic-dispatch mechanism decouples the dispatch method from the class membership. This decoupling lets programmers achieve reuse both by inheritance and reuse in a function.

For more information, contact *davew@cs.haverford.edu.*

Guest Lecture

Extreme Programming: A Lightweight Process

Robert Martin, Object Mentor, Inc.

Robert Martin gave a speech on a popular topic – extreme programming (XP) – and on applying XP in software projects. Ken Arnold defines XP as "a lightweight methodology for small- to medium-sized teams developing software in the face of vague or rapidly changing requirements." XP promotes writing test before code, pair programming, collective code ownership, frequent integration of code base, clean and simple coding style, and frequent communication with customers. Adapting XP practice into your environment improves adaptability, predictability, number of options, and humanity in the development process.

Martin explained that the basic principles of XP are:

- *Rapid feedback* – This facilitates a fast learning environment. You should get immediate feedback for schedule, quality, process, and morale.
- *Assume simplicity* – Solve the problem at hand. Treat every problem as if it can be solved with ridiculous simplicity.
- *Incremental change* – Big changes make things break all at once. Thus, problems should be solved via a series of small changes.
- *Embracing change* – The best strategy for dealing with changes is the one that preserves the most options while solving the most pressing problem. See volatility of requirements as an opportunity, not as a problem.
- *Quality work:* – Everybody likes doing a good job. The only acceptable quality levels are excellent and insanely excellent.

Finally, Martin presented some vivid examples through the perspectives of managers, software developers, and customers to demonstrate how software teams can adapt XP practice.

For more information, see *http://www.objectmentor.com*.

INVITED TALK

Running Through the Woods — A Story About Software Engineering

Bjorn Freeman-Benson, QuickSilver Technology

Dr. Bjorn Freeman-Benson talked about software engineering lessons he had learned from his involvement in several successful large software systems. He made an interesting analogy between software engineering and orienteering. Orienteering is a sport of navigation in which, using map and compass to select routes to run through a series of points shown on the map, one tries to reach the destination in the shortest time. Both software engineering and orienteering have four basic variables:

- *Time* – both want to reach the goal in the shortest amount of time.
- *Cost* – both want to deploy competent tools with minimal cost, e.g., quality of map vs. quality of software engineering tools.
- *Features* – both should avoid using tools with over-complicated features that may adversely affect the progress.
- *Quality* – both care much about the quality of the progress in reaching their goals.

After a brief introduction to orienteering, Dr. Freeman-Benson went on to review four projects in which he had been involved. These projects included Visual Age of OTI, Rose+ of Rational, Amazon's front-end software, and Q compiler of QuickSilver Technology. All rather successfully met their goals as originally set, but in retrospect, he could see things that could have been done differently to make the projects optimally successful.

In conclusion, Dr. Freeman-Benson pointed out that, as with orienteering, in software engineering, you can't build the right thing if you don't know the end goal, if you don't have a plan for how to get there, and if you're sloppy about building it. As the business climate has changed, it has become even more important to build something right instead of just building it quickly.

Session: Reflection in Distribution

The Design and Performance of Meta-Programming Mechanisms for Object Request Broker Middleware

Nanbor Wang and Kirthika Parameswaran, Washington University, St. Louis; Douglas Schmidt and Ossama Othman, University of California, Irvine

Distributed Object Computing (DOC) middleware frameworks, such as CORBA, have gained much popularity in recent years because they shield developers from many complex issues in network programming. There are many meta-programming mechanisms available in CORBA that can further shield the developers from other accidental complexities and improve the adaptability of DOC systems and applications. Kirthika Parameswaran presented a comparison between several of these mechanisms.

Meta-programming mechanisms improve application adaptability by abstracting out certain behaviors into replaceable meta-objects. By using meta-objects implementing different strategies and/or behavior, developers can modify different aspects of a system in a non-intrusive way without losing the maintainability of the application. In this paper, the authors show how the meta-programming mechanisms can be applied at different levels in the ORB invocation path and the challenges of implementing them. They also contrast and compare the scope and implications of applying these mechanisms into applications and provide some guidelines on how to select the proper meta-programming mechanisms.

For more information, contact *kirthika@cs.wustl.edu*.

Kava – Using Bytecode Rewriting to Add Behavioral Reflection to Java

Ian Welch and Robert J. Stroud, University of Newcastle upon Tyne, United Kingdom

When reusing code from outside sources, it is often necessary to modify the original implementations to adapt them to new needs. In this paper, Ian Welch presented another approach to use behavioral reflection to modify the existing programs by rewriting the Java bytecode at code loading time. The behavioral reflection added by the code rewriting associates each object with a meta-object that provides operations like beforeExecution, afterExecution, beforeInvoke, afterInvoke, beforeException, and afterException that developers can use to reflect into the associated operations and object. These operations in meta-objects are invoked automatically through the rewritten bytecode. Developers can use these operations to modify the targeted operation in the original code transparently.

Work on Kava revealed insights on the following topics:

- *Strong encapsulation* – It is difficult to bypass the meta-object bound to the base-object.
- *Inherited methods* – Operations redefined in derived classes will not work with the parent class' meta-object.
- *Exception handling* – Exceptions can be intercepted by the associated meta-object. A context object is used by the meta-object to simplify the meta-object protocol and to allow lazy reification.

For more information, see *http://www.cs.ncl.ac.uk/research/dependability/reflection/*.

Content-Based Publish/Subscribe with Structural Reflection

Patrick Th. Eugster and Rachid Guerraoui, Swiss Federal Institute of Technology, Switzerland

Patrick Th. Eugster presented their work on content-based publish/subscribe implementation. The classic topic-based event service groups messages into topics that are relatively static. Several other publish/subscribe services allow grouping the messages based on the contents they carry. The content-based publish/subscribe service described in this paper utilizes Java's reflection mechanism to acquire the type information of a message. Condition objects are used as filters to define the subscription patterns.

The paper also discusses several performance optimizations, such as avoiding redundant invocations and enforcing static filters. Benchmark results show several metrics, e.g., the number of methods supported by messages, which affect the performance of this approach. These overheads are incurred by the use of Java reflection. They are working on a new optimization that will generate code for static filters based on the execution results from dynamic filters.

For more information, contact *Patrick.Eugster@epfl.ch.*

GUEST LECTURE

Language Integration in the Common Language Runtime

Jennifer Hamilton, Microsoft Corp.

.NET is Microsoft's next-generation development platform that provides easy integration within and across languages and platforms. The Common Language Runtime (CLR) is the fundamental building block of the .NET framework. Jennifer Hamilton gave an overview on the design of CLR and how CLR helps support inter-language object sharing. The major features in CLR include:

- *Common Type System* (CTS) –The single-type system CLR supports. CTS is designed to support multiple languages that .NET implements.
- *Metadata* – Used to used to describe and reference the types defined by CTS. The format of metadata is language independent. Using metadata allows applications written in different languages, programming tools, e.g., compilers and debuggers, and virtual runtime to inter-operate. All objects are strongly typed through the metadata.
- *The Common Language Specification* (CLS) – An agreement between language designers and framework designers. As CTS is too broad for most languages to implement, CLS defines a subset of CTS to guarantee language integration. Exception handling is part of the CLS.

Microsoft Intermediate Language (MSIL) specification defines a type-neutral language that other supporting languages can be translated into, similar to Java bytecode. A unit of object deployment is called an *assembly* and contains self-describing metadata that record the modules and files it contains and the dependencies of containing modules with external modules. The execution model uses a "Class Loader" to link in an assembly.

For more information, see *http://www.microsoft.com/net/.*

Session: Programming Techniques

PSTL – A C++ Persistent Standard Template Library

Thomas Gschwind, Technische Universität Wien, Austria

Thomas Gschwind showed that it is impossible to use a Standard Template Library (STL) container to access persistent data simply by implementing an allocator class because of certain inherent constraints, such as the inability to name permanent data storage through a container interface or query an existing object with the allocator interface. Persistent containers are useful for managing and accessing databases. It is necessary to couple the container interface more tightly with the allocator interface to support persistent containers.

Gschwind described a design that added serialization, deserialization, referencing, and locking capabilities into his Persistent STL (PSTL) containers. He also examined the PSTL compatibility with the existing STL containers and compared the performance of PSTL containers with Berkeley DB and gmdb. He found that with PSTL, it is possible to replace regular STL with PSTL with minimal modification.

For more information, contact *tom@infosys.tuwien.ac.at.*

Making Java Applications Mobile or Persistent

Sara Bouchenak, SIRAC Laboratory, France

Sara Bouchenak pointed out that while Java provides code and object mobility and persistence, Java does not provide any support for mobility and persistence for control flows and execution states. Her work in thread migration provides a framework to stop the execution states of a thread so the thread can be migrated to another machine, or checkpointed on disk for later recovery.

She described the challenges in designing a thread-state capture/restoration service and its implementation. The performance of the service was also documented and showed it to be rather competitive. The paper also discusses several application examples on thread migration.

For more information, contact *Sara.Bouchenak@inria.fr.*

BEAN MARKUP LANGUAGE: A COMPOSITION LANGUAGE FOR JAVABEANS COMPONENTS

Sanjiva Weerawarana, Francisco Curbera, Matthew J. Duftler, David A. Epstein, and Joseph Kesselman, IBM T.J. Watson Research Center

Being able to compose components together dynamically is a vital part of applying component technology. Most programming languages, however, do not treat components as first-class citizens and, therefore, do not provide enough support for component development. Francisco Curbera presented the Bean Markup Language (BML), which supports component composition in a first-class manner for JavaBean components.

BML is an XML-based descriptive language used to describe inter-component binding, construct aggregates of components, and allow macro expansion for constructing certain types of recursive compositions. Curbera's BML implementation includes support for scripting event adapters to bridge components together. They have implemented a framework to compose component aggregates based on a given BML description. A GUI front end is also available to support visual composition of components.

For more information, contact *curbera@us.ibm.com*.

DESIGN PATTERNS FOR GENERIC PROGRAMMING IN C++

Alexandre Duret-Lutz, Thierry Géraud, and Akim Demaille, EPITA Research and Development Laboratory, France

Thierry Géraud presented their work on applying some Gand of Four patterns in generic programming. In generic programming, higher efficiency is achieved by the use of parameterized classes. This paper presents the following patterns: Generic Bridge, Generic Iterator, Generic Abstract Factory, Generic Template Method, Generic Decorator, and Generic Visitor.

For more information, contact *Thierry.Geraud@lrde.epita.fr*.

USENIX Needs You

People often ask how they can contribute to the USENIX organization. Here is a list of needs for which USENIX hopes to find volunteers (some contributions reap not only the rewards of fame and the good feeling of having helped but also a slight honorarium). Each issue we hope to have a list of openings and opportunities.

- The *;login:* staff seeks an acquisition editor for cartoons and other drawings. This paid position, which sounds so simple but turns out to be quite challenging due to the huge number of cartoons out there, should be especially entertaining. Contact *login@usenix.org*.

- The ;login: staff seeks good writers (and readers!) who would like to write reviews of books on topics of interest to our membership. Write to peter@matrix.net.
- The *;login:* editors seek interesting individuals for interviews. Please submit your ideas to *login@usenix.org*.
- *;login:* is seeking attendees of non-USENIX conferences who can write lucid conference summaries. Contact Tina Darmohray, <*tmd@usenix.org*> for eligibility and remuneration info. Conferences of interest include (but are not limited to): Interop, Internet World, Comdex, CES, SOSP, Linux World, O'Reilly Perl Conference, Blackhat (multiple venues), SANS, and IEEE networking conferences. Financial assistance to cover expenses may be available. Contact *login@usenix.org*.
- *;login:* always needs conference summarizers for USENIX conferences too! Contact Alain Hénon *ah@usenix.org* if you'd like to help.
- The *;login:* staff seeks columnists for:
  - Perl
  - Large site issues (Giga-LISA),
  - Hardware technology (e.g., the future of rotating storage)
  - General technology (e.g., the new triple-wide plasma screens, quantum computing, printing, portable computing)
  - Paradigms that work for you (PDAs, RCS vs. CVS, using laptops during commutes, how you store voluminous mail, file organization, policies of all sorts)
  - Comics/cartoons (need to find them, not necessarily draw them).

  Contact *login@usenix.org*.
- The *;login:* staff seeks editors for "special topics" issues or partial issues. If you would like to assemble a collection of articles on a particular topic please contact Rob Kolstad, *kolstad@usenix.org*. (See, for example, the November 2000 issue on Security). This is a paid position.

needles in the craystack: when machines get sick

Part 5: In Search of Cleopatra's Needles

by Mark Burgess

Mark is an associate professor at Oslo College and is the program chair for LISA 2001.

<Mark.Burgess@iu.hio.no>

During his fifty-four year reign around 1500 B.C., the Pharaoh Thothmes III saw erected two 70-foot columns before the great temple of Heliopolis, near what is today Cairo. These two exclamatory masts were powerful symbols of Egyptian spiritual and technological supremacy, singular and immutable signals to contrast with North Africa's seething, inconstant desert sands. For almost 3,500 years, they adorned the entrance to the impressive temple as a symbolic gateway, with little competition from their surroundings.

The first of the needles changed hands as a gift to the British people, in 1819 in recognition of Nelson's victory over the French fleet at the Battle of the Nile in 1798. An odd gift perhaps, a lump of stone so heavy that it took years even to muster the effort to move it – but the significance lay more in its shape than its composition. The symbolic gesture was repeated for the United States in 1881, when the Khedive of Egypt, hoping to stimulate economic investment in his country, gave the twin monument to the city of New York. Since then, the obelisks have been emulated in several large cities, including Paris and Washington. They have nothing at all to do with Cleopatra (preceding her by some 1,500 years), nor are they made of any valuable substance, but their singularity of form conveys a powerful symbolism which is recognized by all cultures.

Symbolism is at the very root of our being. It is about the attachment of meaning to patterns (patterns of material or of behavior). It is a seemingly irrational phenomenon, not unique to humans, but quite possibly the very essence of our intelligence. Symbolism is interesting because it underlines an important dichotomy: the difference between information and *meaning*. In Part 4 of this series, we met the three horseman of entropy: the good, the bad, and the ugly. Information (entropy) has three interpretations: the good (that variation is information), the bad (that variation, hence information, brings uncertainty), and the ugly (that information tends to increase with and result in degenerative aging, or disorder).

The essence of these interpretations is that information and meaning are two very different things. Indeed, too much information is simply noise. Imagine a television from which the antenna has been removed. The screen is a speckled swarm of fuzzy dots, devoid of any obvious meaning. However, it is not devoid of information. On the contrary, the length of a message you would have to send to a friend, so that he or she could reproduce the precise picture, would have to be very great indeed. The point is not that there is little information, but that there is so much information that it is impossible to find any meaning in it. If one could trace the effect back to its many causes, one would find the history of colliding particles in the early universe which led to today's cosmic background radiation. All that information is represented on our television screens, but the essential connection from cause to effect is a little more than we are willing to grasp, or care about.

Failing the Half-Closed-Eye Test

It is necessary to limit information in order to attach meaning. Symbolism is about attaching meaning to limited lumps of information. Lighthouses, totem poles, monu-

ments, company logos, and even icons of the natural world, such as mountains (e.g., Mount Fuji), are all simple statements with very low entropy: concentrated powerful signals which are easily seen against their surrounding environments. The more complex a message is, the more analysis it requires, and the harder it is to discern its meaning.

The importance of such strong signals is not just symbolic, rather the opposite: the reason they have symbolic significance is because they are *effective competition* with their surroundings. Competition is not a word normally used in computer science, certainly not in information theory: we are used to the relative certainty of propositions and logic, not to the bullying and submission of the boxing ring; but competition is a crucial concept whenever systems interact.

This has broad ramifications for information. Strong signals have a powerful effect on the environment. Conversely, too much information or meaning becomes garbage. It is indistinguishable from no information. Indeed, the whole concept of "renormalization" in science is about trimming away the noise to see what is left. If every signal is equally loud, the result is just loud noise.

Look at the Web pages of many ISPs, for instance, and one finds a seething jumble of signals, from advertisements to links, to imagery, to text and color – not unlike the pornographic neon jumble of urban market districts. If one half closes one's eyes and looks at a page of information, the immediate impact of its layout and any strong symbols is all that can be seen. Today, multimedia messages assail us from every angle, detracting from overall impact of any one part. Commercial television is perhaps the worst example in some countries, where invasion by commercials, rapid cuts, overlaid text information, unnatural emphasis of voice, inappropriate music and never a second of silence, pollute any message being conveyed with toxic irrelevancy.

To make a garbage heap, one only needs to collect a sufficient number of individually meaningful things and put them all together. There is a reason why the Washington Monument (another obelisk) is used as a symbol of strength and unity, rather than a haphazard slum or shanty town, where there is much more information. There is a reason why Big Ben in London is well known; the Vigeland Monument in Oslo; the Eiffel Tower in Paris; the Statue of Liberty in New York. These are strong signals, dominating their environments.

Today, we constantly erode the meaning of symbols by abusing them out of context. Think of mobile phones which now spam us with well-known music, rather than the low-info bell, or imagine the Eiffel Tower in the Amazon rain forest. Such loss of context demeans the significance of symbols, creating garbage out of art. Understanding the significance of mixed signals is a subtle business; in fact, it is one of the perpetually unsolved problems which demands our attention. Part 4 of this series was about how to use information maximally, by limiting and structuring it; it was also about the fundamental limitations incurred when information is limited for the purpose of ordering. That discussion provided a mapping from cause to effect and showed the limits incurred on going back the other way.

But what about this opposite direction? Separating signal from noise is much harder. Going from observed effect to the possibly many causes is a much harder problem, because summarial effect is often a mixture of many causes (a many-to-one mapping), and the combination is not necessarily a linear superposition. Yet this problem is clearly at the heart of all diagnostics, fault analysis, intrusion detection, and performance tuning. What can we hope to find out from observations?

To make a garbage heap, one only needs to collect a sufficient number of individually meaningful things and put them all together.

For most things, random means "too complicated to really analyze."

"Imagination is more important than knowledge"

We take many complex things around us for granted. For instance, according to statistics textbooks, coin tosses and rolled dice are considered random events. Is this true? A coin has only two sides, it is circular: the simplest shape possible in two dimensions. A die is scarcely more complicated. Isn't it possible to predict the outcome? How hard could it be?

What we know is that, if we do toss coins or dice, the distribution of results, over many throws, seems pretty random. The reason is, of course, only that we can't be bothered to figure it out. It is not all that difficult, *in principle*, to predict the outcome. We know Newton's laws, we know about gravity, and we know about geometry. So what's so hard? The answer is: everything else – the environment. The environment (fingers, the table, floor, air, height, etc.) comprises a bunch of variables which are quite complicated. Since we are lazy, and the problem of coin tossing is not the least bit interesting, we pretend we don't know how to do it and call the result "random." So random means "other variables" which we don't account for. It turns out that, when we get down to quantum mechanics, and the subatomic, the world turns out to be unpredictable for completely unknown reasons, but for most things, random means "too complicated to really analyze." In a coin toss, we classify all of the complex variables into just two outcomes – a many-to-two mapping.

If we want to know what causes the result, it is not possible to extract information about all of those causal variables just from the two outcomes. One cannot reverse a many-to-few mapping. Lost detail is gone forever.

Statistics was invented in order to get something out of situations like this. Statistics says: "Even though we can't reasonably analyze some problem in complete detail, we can still try to say something about them." The usual way that statistics is presented is rather obtuse, and even a little dishonest. The idea is that we go out and measure a bunch of stuff in some problem and plot the outcomes in some way. Then we try to fit some off-the-shelf probability model (Gaussian, Poisson, Parot, etc.) and work out a bunch of standard things. That approach is a little bit like trying to say something about human behavior by matching hairstyles.

In order to use statistics meaningfully, we need a model of what causes the results: a hypothesis which can be tested. That means we essentially have to guess how the mapping from many causes to few results works. From such a model, one can then work out the consequences and see whether they match observation. This is how science is done. Einstein, always good for a quote, said that "imagination is more important than knowledge." What he meant was that, to understand nature, we need to dream up models by imaginative means. They can then be confirmed or denied. He did not mean, as some have suggested, that imagination overrides knowledge, only that knowledge itself is not enough for finding answers: creativity is needed.

Once information has been discarded or lost, it is not possible to go back and recreate it, unless it is somehow coded into a local potential, as was discussed in Part 4, or journaled in an ever growing archive. The only way to refine one's understanding of events, to amplify on "it was just random," is to postulate a reason and then collect evidence which supports or denies it. Imagination is itself just a random entropic walk of possibilities, loosely constrained by a hypothesis.

A Comedy of Errors

Our brains are extremely good at fitting models to data – almost too good, in fact. It is as inconceivable as it is inevitable. When I was younger, I would sit and watch the television noise, after everyone had gone to bed, with a friend. While listening to Led Zeppelin, we would wait for the rabbit on the bicycle to race across the bottom on the screen, as it often did, late at night. The rabbit had apparently been observed by many of our friends, while watching the fuzzy dots. Of course, some prefer to look for faces in clouds, or emotional expressions on car radiator grills. The essence is the same.

At the end of the 19th century, the Italian astronomer Sciaparelli was having much the same problem. He trained his telescope on Mars and was amazed to find a criss-cross pattern of lines which he called *canali*. Percival Lowell misinterpreted his notes and thus began the legend of the Canals of Mars. The canali were later discovered to be a trick of the low-resolution distortion, or a loss of information, about random dark spots on the surface. A century later, another blurred picture of Mars apparently revealed a gigantic human face carved into the surface. On closer inspection, it was another trick of the light, fueled by lack of resolution, much to the disappointment of UFO enthusiasts and indeed Hollywood, who went ahead and made the movie anyway.

Model fitting is intrinsic to the way our cognition works, and we use that to good effect. When we look for meaning, we do so by attaching interpretive models to data we perceive. The process of problem solving is the forward process of mapping cause to effect. The reverse process is that of diagnostics, mapping effect back to cause. Our capacity for reasoning might even have developed as a by-product of this ability for causal analysis.

Models are essential to the way we think, so to understand any data, to diagnose any situation, we need some kind of causal model. We build mental-model imagery by generating high-entropy associations, when we parse language; this gives us the ability to infer things from previous experience and leave things unsaid. When we say "cake," we don't just think of a description of physical attributes, we think of the cake Granny made on Saturdays when we were small. We think that it was like the one in the cafe the other day. We think, not of one cake but of all cakes, of different colors, shapes and sizes, and of all our experiences eating cakes, and of parties. Perhaps we think of a cookery program on the television or a famous chef. From one starting place, our thinking diffuses into every niche of possible meaning. Those who are good at diagnostics are often those who are good at this random walk of association.

In other words, concepts are not isolated things in our minds. The robustness of meaning has only to do with the competitive strengths of the different interpretations. Our memories are dense networks, tied into many models. We have no unique search-key: there is no unique way of attaching meaning to memory; rather, we tie memories into many different models of meaning, which compete for attention. This feature of human cognition is what we exploit in blending systematic, rational thought with the apparently random walk of imagination. It is how we solve problems and diagnose causal change. This is not at all like databases. Computers work more like warehouses. We find a piece of information and store it on a shelf at a particular place. None of the other information is aware of it. No free relationships are forged. We have learned to make simple associations with relational databases, but these are very primitive.

Time's Causal Arrow

One of the apparent paradoxes of change, which was first pondered in the world of physics, is how the apparently reversible laws of physics can lead to irreversible conse-

Model fitting is intrinsic to the way our cognition works, and we use that to good effect.

Complex changes are not easily undone without a memory of exactly what transpired.

quences. Reversibility is a property of physical law which means that time has no arrow. The fundamental equations of physical systems work just as well forwards as backwards. There is no concept of past or future. This resembles the problem of correlation in statistical analysis. If two things are correlated, A being correlated with B means that B is correlated with A. There is no arrow which tells us whether A caused B, whether B caused A, or whether they are both by-products of some deeper cause. This is precisely the problem that models try to unravel.

The "paradox" of reversibility is not really such a mystery to the three horsemen of entropy, because it all has to do with how things spread out into an environment. Many apparent paradoxes of physics arise because physicists forget the environment surrounding their object of attention, as a matter of routine. Physics is about isolating the signals of nature into single threads which are independent of one another. The aim is to find the arrow from cause to effect as clearly as possible, by stripping away the jungle of irrelevancy. Amongst the approaches used is to try to physically isolate effects from other signals by putting them in a box, or by shielding systems from their environments, or by performing control experiments and subtracting one from the other to obtain the variance.

The mystery of reversibility is this: the laws of physics describe infinitesimal changes; they are formulated in terms of "differentials," or changes so small that the environment is irrelevant to them. However, the laws also include recipes for combining small changes into large ones. It is at this larger scale, where we take a step back from the small details and do the half-closed-eyes test, that the environment becomes important. How we combine the infinitesimal changes is important, and that combination puts out feelers into the environment, where other signals and effects are lurking. Every time we combine tiny changes into larger ones, the effects of the environment play an infinitesimal role. As we get farther and farther from the starting point, and after many such combinations, we begin to see a real change, reflected in the landscape of influences from the surroundings. Systems spread out into their environments, mixing with other signals as they go. Suddenly, the way the infinitesimal changes were combined (their history) becomes very important to the final result.

The following analogy might be helpful. If you drive your car one meter in any direction, the world around does not have any great effect on you, and the change is easily undone. However, if you drive half way across the continent, then the route you take begins to play an important role: the features of the landscape make one route unequivalent to another. That places an implicit arrow on the journey. Complex changes are not easily undone without a memory of exactly what transpired. The way that physics is formulated, the memory of how changes transpire is usually lost (dissipated) to the environment; one chooses to ignore the information and treat it as ambient noise, and thus it seems like a mystery how the rest of the changes happened.

When we look for changes in the machinery of computer communities, this principle is of central importance. It is not enough to collect data about changes, do little statistical analyses, plot graphs, etc. if one cannot separate the important signals from the environment. Physicists throw away parts of a signal which they are not interested in (and sometimes get confused, but not as often as one might expect); this is how they find order in chaos.

The meaning of signals is a mapping from cause to effect. Signals which change things are more than just correlations (which are bi-directional), they are directed arrows, like

conditional probabilities. Auto-correlations have direction in a time-series only by virtue of time's arrow.

Pins and Needles

When machines get sick, they exhibit certain observable symptoms. These are clues as to the cause. Humans, for instance, get sore throats, perhaps skin coloration, headaches, pain, and so on. The same symptoms characterize most illnesses because there are only so many things which can hurt or change visibly. Computers run slowly, perhaps even stop working or jumble data. There is only a finite number of symptoms which we observe, but the number of possible causes is far greater. In a sense, finding out the cause of machine illness from a few symptoms is like the problem of determining why a flipped coin shows heads or tails. It is a many-to-one mapping, which one is trying to reverse.

Of course, we would like to correct an illness at its source, if possible. Merely addressing the symptoms ignores the causal chain of events which led to the problem, and the fact that the chain is often unidirectional. Turning over a tossed coin does not change the conditions of the environment which selected one of its faces in the first place. The level of detail in such a response would be incommensurate with the level of detail in the environment which caused the result. That cannot be a cure. Similarly, patching symptoms does not cure the illness which causes them.

In complex machines, the cause of sickness has to be a strong signal. There is a lot of complexity, or entropy in modern computers, in biological systems which compete for resources. A weak signal would just be a whisper in the wind. In order to be noticed amongst the other things going on, the problem has to be sufficiently strong. That often means that by the time the signal has grown to noticeable levels, it is already well established, and hard to counteract.

The significance of such a signal would be its strength. Humans get sick when bacteria or viruses replicate themselves to such a degree that they present a signal in our bodies which is so strong as to be toxic. They start drawing resources from other tasks which suffer as a result. The point is not whether they are foreign or not. Cancers are not foreign, but they are also strong signals which are pathological. The effect of a strong dose of almost any substance or signal (even water!) is toxic to a system, because it drives that system in a direction which is not its usual one.

The Search for Extra-Network InteLligencE?

In recent years, it has become popular to build anomaly and intrusion detection systems that listen for rabbits on bicycles amidst the storm of traffic on networks. What is the likelihood of finding anything interesting?

We would like to be on the lookout for signals which could be dangerous or helpful to us. There are many signals out there, waiting to be understood. Searching for messages in complex signals is like trying to find a needle in a haystack. There are many examples of this problem: all kinds of diagnostics, fault detection (including medicine) may be viewed as such a search. Looking for genes in DNA by examining the coding and looking for patterns is another example. Even more difficult is the problem of figuring out the causal relationships between gene action and manifestations of phenotype (species and characteristics) and cell function. In the Search For Extra-Terrestrial Intelligence (SETI), scientists look for what might be a meaningful signal in the bath of fuzzy dots on your TV screen. Cryptanalysis has many of the same difficulties: how to tell a bitstream encrypted message from noise?

Searching for messages in complex signals is like trying to find a needle in a haystack.

In system administration we are often trying to walk the fine line between necessary complexity and the ability to control.

In each case, the problem is to identify meaning in a mass of data by looking for causal threads. If we were lucky, every important signal would be a strong, low-entropy obelisk contrasting with a sandy desert. Unfortunately, nothing is so simple. Where you find one obelisk, others appear. Often, for the sake of efficiency, one compresses signals into as small a space as possible. This is certainly true of network traffic. In a desert full of obelisks, none of them seem to distinguish themselves anymore. This, of course, is one thorn in the side of intrusion detection systems. How do we distinguish a good signal from a bad signal? How do we attach meaning to patterns of traffic? Can we, for example, see the difference between a denial of service attack and a legitimate transfer of data? Stealth attacks are deliberately low-entropy signals, designed to be hidden.

In system administration we are often trying to walk the fine line between necessary complexity and the ability to control, but as we have seen in the previous issues, complexity cannot really be controlled, it can only be regulated by competitive means. Sometimes ecologists introduce a species of animal or plant into an ecology, perhaps a predator which will control a parasite or infestation (like the snails or hedgehogs of Hawaii). They introduce one species because that is a simple signal, something which seems to be controllable. Sometimes this happens by accident (as with the grey squirrel in England). In order to make a difference, such a species needs a selective advantage, which is a very low entropy signal. The problem with such a signal is that it will tend to dominate. Dominant signals are often toxic. On the other hand, if one were to just throw a bunch of random animals into a system, their effect would be unpredictable, but perhaps more balanced. This is the dilemma discussed in Part 3.

Another example is drugs. Drugs (medications) are low-entropy signals designed to target specific "problems," where problem is defined as something possibly dangerous or undesirable to us. Drugs are failing today because they make themselves very obvious. If you keep hitting something in the same place, it will either wear out or move out of the way. A good analogy would be that, if the Germans had invaded Poland and kept on invading Poland during the Second World War, then everyone would simply have moved somewhere else and left them to it. That would not have had the desired effect. Had they attacked everyone at the same time, the result would probably have been a failure, because each signal would have been too weak to be noticed. ("You, soldier! Surround the enemy!")

When the environment or an external influence acts with a strong signal, it leaves a marked effect and tends to order the system. This is the way that certain genes become important. This is the way that computer programs are able to learn by finding signals which can be digitized into a particular path through a data structure or a digital value. This is how the brain learns, we presume, although no one actually knows how the brain learns. Bacteria become resistant to drugs by genetic selection in the face of a low-entropy signal. A single threat is easy to resist. Too much of a good thing, and it loses its significance.

Rotate Shields; Long Live the Collective!

Machines are subject to a variety of influences. Every influence is a signal, a message to the system which changes it somehow. Some changes are good, some bad, and some neutral. This bath of information is called the environment. It acts like a potential, or fitness landscape in which a machine is meant to survive and fulfill its function. Those machines which have evolved in complex environments have evolved defenses to many of the signals which are contrary to their stability. Human-made machinery, however, is put together in the way that humans see the world: by the half-closed-eye method. We

extract the simplest machine which solves the main part of a complex task. We build to solve the problem as we see it.

Simplification helps understanding, but to quote Einstein again: "Everything should be made as simple as possible but no simpler!" If animal evolution had simplified to the extent that human machine-building does, they would all be dead. Human machines have traditionally been built for protected environments: assuming that the user will do no harm. Today, the network has unleashed a force which has taken the propositional world of computing by surprise: *competition*, fuelled by its ally *diversity*. Entropy arrives center stage, and our machines are ill-equipped to deal with it. Our diagnostic abilities have not been built into our technology in the same way that evolution pits competition against strong signals.

Once machines, driven by humans, were connected in a network, it was inevitable that there would be a variety of players with a variety of motivations, some good, some bad, and some ugly. The environment of users and other machines, which our computers bathe in, by console and by network, contains many signals which help and oppose the system. We are just learning how to shield ourselves from such signals, as DNA did with cell walls, as organisms did with skin: we can build firewalls and filters for the network and access controls for resources, so that unwanted signals are not passed.

These controls are not perfect, however. The environment is huge, the machine is small; the resources available to a complex environment, to prod and to poke the system for weakness, are huge. Unlike, genetically determined machines, computers are reliant on humans to make changes. Adaptive systems do not really exist today.

Strong signals are needles in the side of the machine. There are a few ways that machines can resist such toxic signals. The first is to have a defense of any kind. The next level is by redundancy: if one strategy for protection fails, another can take over. The next is the bane of the Borg: rotating shield modulations. We vary the defensive strategy constantly, to prevent an attacker from adapting to the defensive signal. This is the strategy used by screen-savers, to avoid permanent damage to monitors from persistent, strong signals. In a competitive environment, constancy is both your friend and your enemy. It is an arms race, i.e., a race to stand still. It is the principle of the Red Queen (a quote from *Through the Looking Glass*):

> *"Well in our country," said Alice, still panting a little, "you'd generally get to somewhere else – if you ran very fast for a long time as we've been doing."*
>
> *"A slow sort of country!" said the Queen. "Now here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run twice as fast as that!"*

In computer systems, one could rotate system policy at random (cfengine can do this, for instance) to prevent users from learning system policy and adapting to it. By making the signal less obvious, one can actually win stability. This is the lesson of non-cooperative Game Theory. In the future, we shall have to learn to adapt and embrace complexity, if we are to create machines which have the ability to communicate in a collective. If we haven't learned this already from society (love thy neighbor and lock thy door), then we'll have to learn it all over again, the hard way.

The Babel Acupuncture

Our world is teeming with information: its jungles, its species, its cities and the ever changing dynamics of it all are an information system of formidable complexity. From

"Everything should be made as simple as possible but no simpler!"

this complex world, humans have imagined meaning in the simplicity of low-entropy symbols. It is a Tower far greater than Babel. Then, having built the Tower, we push it down again, polluting precious meaning through repetitive strain, and the meaning becomes the garbage which we fail to tidy up after ourselves, like a multimedia virus which sweeps to every quiet corner of the world. The result might be something indistinguishable from the fuzzy snow on a TV screen. The "Snow Crash" is no myth.

All the needles we have made, we turn on ourselves, in our community bartering grounds, as we compete for supremacy in whatever value system we hold dear. Whether economist, hacker, warmonger, or merely egoist, we slowly shoot ourselves in the foot with tiny needles.

palm pilots and the point-to-point protocol

by James Caple

James Caple has eight years of combined industry experience in proposal development, hardware hacking, UNIX systems administration, and most recently, Java and C programming.

jcaple@patriot.net

Sun Microsystems is right: "The Network Is the Computer"™. Chances are, if you use a computing machine of any sort, you have a need to make it network-aware at some point. Personal Digital Assistants (PDAs) are great candidates for both wired and wireless networking, extending the functionality of your networked desktop workstation to your pocket. This article discusses a simple set of steps for establishing a persistent connection between your Palm Pilot and your Linux workstation using Palm's Hot Sync cradle and the Point-to-Point Protocol (PPP).

Note: Please refer to RFC1661 for further details regarding the PPP.

So What?

Why would you want to do this, you ask? Well, I am not sure I can answer that for you, but here are some reasons that might help:

1. You are writing a networked Palm application that you want to test on a real device, but you don't want to pay for a wireless modem just to test your application.
2. You need to telnet to a remote workstation and all you have with you is your Palm Pilot (it could happen . . .).
3. You want to set up a kiosk consisting of nothing but Palm cradles and a Linux box where you want users to be able to browse the Web using only their Palm Pilot.

These are some possible situations where the knowledge presented in this article might be of some use. If for no other reason, it is an interesting exercise nonetheless. Furthermore, it is kind of cool to have the ability to telnet or browse the Web using your Palm Pilot.

For those of you most familiar with Microsoft Windows, the subject of this article may seem less arcane since using the Microsoft RAS protocol makes PDA networking rather simple. It's not as evident how one would go about networking a Palm Pilot with one's Linux workstation, however. Recognizing that wireless Palm Pilot networking, using a wireless modem (like those offered by OmniSky) or a Palm VII is optimal for PDA networking, these alternatives are still rather pricey. PPP networking provides a cost-effective alternative for doing cool networking "things" with your Palm Pilot: application testing, for example.

Preparation

Most Linux distributions come with a number of PPP utilities. If you are unfamiliar with PPP, please refer to the PPP HOWTO articles at *http://wwww.linuxdoc.org*. Establishing a PPP connection between your Palm and Linux workstation is not that different from establishing a PPP connection with your Internet Service Provider (ISP), although configuring your Palm device correctly can be a little tricky. Also note that in the steps presented below, I am using the Beta version of RedHat Linux 7.0 (kernel version 2.4.0-0.99.11), and a Palm Vx (Palm OS v.3.5.0), so your results may vary slightly depending on your specific configuration.

OK, here we go. First, connect your Palm cradle to a free serial port at the back of your workstation. Make sure your cradle is properly connected. One way to do this is to grab

Step 1

Step 2

Step 3

pilot-link from *http://pilot-link.sourceforge.net* and try to use the pilot-xfer utility to transfer a Palm Database (PDB) from your device to your workstation. For example, try to transfer MemoDB as follows:

```
[root@localhost]# pilot-xfer /dev/ttyS1 -f MemoDB
```

If you have trouble with this step, try using /dev/ttyS3. If all else fails, read the pilot-xfer documentation for further details.

PPP Configuration

Secondly, if you want to be able to reach machines beyond your own Linux workstation with your networked Palm Pilot, you will need to make sure that your kernel supports IP Masquerading. In the worst case, you will need to rebuild your kernel with support for IP Masquerading, but chances are, if you are using the latest RedHat distribution, it is already supported in your kernel. In addition, I had to remove the ipchains module in order for my ppplogin script (see below) to work. You can do this using the following command:

```
[root@localhost]# rmmod ipchains
```

You must either create or modify your existing /etc/ppp/options file to contain these parameters:

```
lock
debug
noauth
crtscts
```

Now create a simple script in /etc/ppp called ppplogin, which should contain the following lines:

```
# Load the NAT module (this pulls in all the others).
modprobe iptable_nat

# Note: Change eth0 to ppp0 if you use a modem.
# Dial-up for your Internet connection.
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# Turn on IP forwarding.
echo 1 > /proc/sys/net/ipv4/ip_forward

# 10.100.166.56 is the IP of the local Linux workstation.
# 192.168.1.2 is the IP I wish to assign to the Palm device.
# Modify accordingly.
/usr/sbin/pppd 10.100.166.56:192.168.1.2 connect \
  '/usr/sbin/chat -vf /etc/ppp/chat' /dev/ttyS1
```

Finally, the last step in configuring PPP on your Linux machine is to create the /etc/ppp/chat script. This file should simply contain a line that reads TIMEOUT 30, or whatever timeout value you wish to assign to pppd.

Configure Network

Now that we have configured the PPP Server, we need to set up the network configuration on the Palm device. The accompanying screenshots, grabbed from the Palm OS Emulator for UNIX, show how to configure your network settings.

Step 1. Tap Prefs and select Connection from the pull-down menu.

Step 2. Select Direct Serial and tap Edit.

Step 3. Tap Details and configure as seen.

Step 4. Select Network from the pull-down menu and create a new Service as shown.

Step 5. Tap the Details button in step 4 and configure as shown.

Step 6. Tap the Script button in step 5 and configure as shown.

While these screenshots may prove helpful, I have made this process somewhat easier by providing Connection and Network Palm Databases containing this configuration information. If you want, you can download these files and install them on your device, which could save you some frustration (see Resources, below). Once you have obtained these Palm Databases, you can use the pilot-xfer utility on your Linux machine to install them on your Palm device like so:

```
[root@localhost]# pilot-xfer /dev/ttyS1 -i NetworkDB ConnectionDB
```

You might want to make a backup copy of your original databases before doing this, however (see the pilot-xfer man page).

Now we are ready to test our configuration. Place your Palm Pilot in the Hot Sync cradle. Tap the Prefs icon and select Network from the pull-down menu. Change your working directory on your Linux workstation to /etc/ppp, and run the ppplogin script as root. You can look at your system logfile after running ppplogin to make sure it is running properly (e.g., tail -f /var/log/messages). The following lines from /var/log/messages indicate that pppd is waiting for a ppp connection request on /dev/ttyS1:

```
Apr 30 22:07:02 velocis pppd[2715]: pppd 2.4.0 started by root, uid 0
Apr 30 22:07:03 velocis chat[2716]: timeout set to 30 seconds
Apr 30 22:07:03 velocis pppd[2715]: Serial connection established.
Apr 30 22:07:04 velocis pppd[2715]: Using interface ppp0
Apr 30 22:07:04 velocis pppd[2715]: Connect: ppp0 <—> /dev/ttyS1
```

Finally, tap the Connect button on your Palm (see step 4, above), and wait a second for the connection to get established. Provided you have IP Masquerading and your PPP settings properly configured, you should be able to access remote machines using a telnet client, or even a Web browser. You can also use this configuration to test your own custom network-aware Palm applications.

Conclusions

Establishing a PPP connection between your Palm and Linux workstation is just about as easy as dialing up your local ISP. There can be a number of benefits in having the ability to do this as well. The most notable benefit I see, however, is that this configuration, using your Palm Linux workstation and Palm Pilot, can provide a low-cost alternative to testing your network-aware Palm applications on real devices, without requiring more expensive wireless modems and service providers.

Resources

RFC1661 – *http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1661.html*
PPP-HowTo – *http://www.linuxdoc.org/HOWTO/PPP-HOWTO/index.html*
Pilot-Link – *http://pilot-link.sourceforge.net*
Palm OS Emulator (POSE) – *http://www.palm.com/devzone*
Sample Palm Network / Connection Databases –
http://www.trexlabs.com/USENIX/pdbs.tar.gz
Telnet Client for Palm – *http://www.mochasoft.dk*
Web Browser for Palm – *http://www.ilinx.co.jp/en*

Step 4

Step 5

Step 6

the tclsh spot

by Clif Flynt

Clif Flynt is president of Noumena Corp., which offers training and consulting services for Tcl/Tk and Internet applications. He is the author of *Tcl/Tk for Real Programmers* and the *TclTutor* instruction package. He has been programming computers since 1970 and a Tcl advocate since 1994.

<clif@cflynt.com>

The last several Tclsh Spot articles have shown ways to write simple client-server applications with Tcl. These pairs all used pure ASCII text to transmit data, in the tradition of SMTP, NNTP, POP, etc. This article will expand on the client-server articles and the previous article about the binary command, to develop a program to examine the behavior of telnet servers.

It's almost embarrassing to admit how many years I've used telnet without ever wondering what went on under the hood. The telnet protocol not only handles the printable ASCII text that we see on our screen, it also sends a number of unseen commands to negotiate how the session will be managed.

Peter Burden published a nice overview of the telnet protocol at *http://www.scit.wlv.ac.uk/jphb/comms/telnet.html*.

The telnet configuration commands all start with a binary 0xff byte, referred to as IAC (for Interpret-as-Command). The 0xff is followed by a binary command between 0xf0 and 0xfe, which is followed by the data required by the command.

Opening a connection to a telnet server is easy in Tcl: we just use the normal socket command.

```
set connection [socket $Client(ipAddress) 23]
```

The default behavior for the socket command is to open a buffered connection to the remote host and translate carriage-return/newline pairs to whatever the normal line termination characters are on the local host.

This works fine for printable ASCII text, but creates a problem with binary data, where a 0x0a might be not be a newline and adding a 0x0d will break a binary data stream.

The solution is to use the Tcl fconfigure command to modify the socket's behavior to something more acceptable for binary data.

The fconfigure command will modify several aspects of the channel including whether the channel should be buffered, the size of the buffer to use, whether to block on read/write and how to translate binary characters.

Syntax: fconfigure *channelId ? name? ?value?*

fconfigure	Configure the behavior of a channel	
channelId	The channel to modify	
name	The name of a configuration field which includes:	
	-blocking *boolean*	If set **true** (the default mode), a Tcl program will block on a gets, or read until data is available. If set **false**, gets, read, puts, flush, and close commands will not block.
	-buffering *newValue*	If *newValue* is set to **full**, the channel will use buffered I/O. If set to **line**, the buffer will be flushed whenever a full line is received. If set to **none**, the channel will flush whenever characters are received.
	-translation*{inMode outMode}*	Controls how end-of-line translations are performed. Acceptable values for *inMode* and *outMode* include:
		auto On input, treats any combination of cr and lf as line terminators. On output line termination is sent as whatever is native for the current platform.
		binary No end-of-line translations are performed.
		cr On input, all cr characters are converted to newline. On output, newlines are converted to a cr character.
		crlf On input, all crlf characters are converted to newline. On output, newlines are converted to a crlf sequence.

We can modify the behavior of our socket with this code:

```
fconfigure $connection -translation binary -buffering none -blocking 0
```

The previous client-server pairs used the Tcl gets command to read a line of text from the server. Again, looking for a newline doesn't work with binary data.

The Tcl read command handles this problem.

Syntax: read *channelID ?numBytes?*

read	Read a certain number of characters from a channel.
channelID	The channel to read data from
numBytes	Optionally, the number of bytes to read. If this argument is left out, all available data is read.

These commands enable us to build a non-line-oriented client with commands like this:

```
set s [socket $argv 23]
fconfigure $s -translation binary -buffering none -blocking 0
fileevent $s readable "readData $s"
proc readData {s} {
    set n [read $s]
    if {[eof $s]} {
        close $s
    return
    }
    # Process data.
}
```

Whenever there is data available in the input buffer, the readData procedure will be evaluated to read and process the data. Since the socket is configured to be non-blocking, the read command will read whatever is in the buffer, whether it ends in a newline or not.

This brings us to the processing part of this example, which is to print out a human-friendly version of the server's requests.

Since Tcl is (like most of us) primarily text oriented, the first thing to do with the binary data is convert it into printable ASCII to make it easier to work with. Since the telnet protocol is byte oriented, it makes sense to convert the data into single byte hex values.

We can use the Tcl binary scan command to convert the binary data to a string of Hex values.

Syntax: binary scan *binaryString formatString arg1 ?varName1? ... ?varNamen?*

binary scan	Converts a binary string to one or more printable ASCII strings
binaryString	The binary data
formatString	A string that describes the format of the ASCII data
*varName**	Names of variables to accept the printable representation of the binary data

For example to convert a string to hexadecimal values:

```
# Convert "abc" to "616263", save value in hexData
  binary scan "abc" "H*" hexData
```

That's better, but still hard to read and work with. If the string were a list of individual byte values, we could read the string more easily and use Tcl's list-processing commands to step through the data.

The Tcl split command will convert a string into a list, splitting on whatever character (or characters) you desire.

Syntax: split *data ?splitChar?*

split	split *data* into a list.
data	The data to split.
?splitChar?	An optional character (or list of characters) to split the data at

A common use of the split command is converting some character-delimited string (like a spreadsheet export file) into a proper Tcl list.

```
# Convert "3/23:Cab from Airport:35.00"
# to
# {3/23 {Cab from Airport} 35.00}
set line "3/23:Cab from Airport:35.00"
set lst [split $line ":"]
```

By default, the splitChar is a whitespace. You can define any other character or characters to be the split character, as the previous example does. You can even declare the split character to be an empty string, which splits a string into a list where each character is a separate list element.

```
# convert "abcd" to {a b c d}
set lst [split "abcd" ""]
```

The next step is to convert the list of single hex values into a list of byte values. We could step through the list two elements at a time using Tcl's foreach command, but since this is a fairly simple pattern to convert, it's faster to use the regsub command.

The regsub command will modify patterns that match a regular expression to another pattern. The syntax looks like this:

Syntax: regsub *?options? expression string subSpec varName*

regsub	Copies *string* to the variable
varName.	If *expression* matches a portion of *string* then that portion is replaced by *subSpec*.
options	Options to fine-tune the behavior of regsub may be one of:
	all — Replace all occurrences of the regular expression with the replacement string. By default only the first occurrence is replaced.
	-nocase — Ignores the case of letters when searching for match.
	— — Marks the end of options. Arguments which follow this will be treated as regular expressions, even if they start with a dash.
expression	A regular expression which will be compared to the target string.
string	A target string to which the regular expression will be compared.
subSpec	A string which will replace the regular expression in the target string.
varName	A variable in which the modified target string will be placed.

In the simple use of regsub, the *subSpec* value will be a simple string:

```
# Convert powerful to useful.
regsub "power" "Regular expressions are powerful"  "use" string2
```

The regexp command will sort parenthesized subsets of a regular expression pattern into separate variables. Similarly, the Tcl regsub command supports merging patterns

from the original string in the subSpec value. These substitutions are specified by using a backslash-escaped number in the subSpec string to correspond to the parts of the pattern you wish to substitute. The number denotes which part of the pattern will be substituted for the backslash-escaped value.

The pattern \0 will substitute for the entire matched portion of the string. If sections of the regular expression are placed in parentheses, these sections can be addressed as \1 for the first parenthesized expression, and \2 for the second, etc.

```
# Invert the positions of "a" and "c"
# x will contain the string : "abc becomes: cba"
regsub "(a)(.)(c)" "abc" {\0 becomes: \3\2\1} backwards
```

Which, finally gets us to a simple three-line procedure to convert a binary string to a list of hexadecimal-coded byte values:

```
# Convert binary strings to lists of hex values
# "abc" converts to "61 62 63"
proc bin2hex {binaryString} {
    binary scan $binaryString "H*" hexData
    regsub -all {([^ ]) ([^ ])} [split $hexData ""] {\1\2} hexList
    return $hexList
}
```

The final process is to step through the list of binary data and print out something a simple human can easily read. The first byte of any command will be the IAC symbol, 0xFF. The value after that will be a byte that denotes which command this is.

In C, we might parse the second byte with a case command like this:

```
switch (i) {
    case FE:  {processFE; break;}
    case FD:  {processFD; break;}
}
```

In Tcl, we can let the interpreter handle the parsing, and instead of using a switch statement to parse our commands, we can name the procedures for the command name.

```
# Define procedures to process the command
proc FE {...} {
    # Process the FE command
    ...
}
proc FD {...} {
    # Process the FD command
    ...
}
...
# Select the command from the hexadecimal list
# (something like FA of FD)
set cmd [lindex $hexList $position]
# Evaluate the command
$cmd
```

The number of bytes after the command byte varies for each command. Some of the information commands (like terminal type) can even include free-form ASCII data.

In C, it would be nice to use a pointer, and advance the pointer as we process each command and data.

Unfortunately, this isn't an option in Tcl, so we'll have to use the lindex command to select each byte we want to look at from the list, and increment our position counter after we process each command.

To make life simpler, the procedures that process each command are defined to return the number of bytes to increment the pointer to get to the next IAC marker.

The telnet protocol allows interleaving of ASCII text and IAC commands in a data stream, so we have to check each byte to see if it's an IAC symbol and process the command if it is.

```
set lst [binaryStrings::bin2hex $string]
for {set i 0} {$i < [llength $lst]} {incr i} {
    # Get the current character
    set l [lindex $lst $i]
    # Loop for as long as the current character is
    #  an IAC marker
    while {[string match $l "ff"]} {
        # Get the command byte
incr i
set cmd [lindex $lst $i]
# Evaluate the command, and increment
#  the position marker
incr i [$cmd [lrange $lst $i end]]
# Get the next byte – if it's still an IAC we'll
#  loop.
set l [lindex $lst $i]
    }
}
```

The procedures that process the commands contain the logic for handling each telnet command and also save a comment about what they've done.

The basic format of the procedures is:

```
proc fd {sublist ...} {
    # Process the fd command
    # ...
    addInfo $subl TELOPT_
    return $commandLength
}
```

The addInfo procedure uses an associative array as a lookup table to get the human-readable information about each command and puts that string into a temporary comment buffer, which is combined by the main loop when a command has completed.

```
proc addInfo {subl prefix} {
    variable Telnet
    variable Lookups
    set m [lindex $subl 0]
    scan $m %x dec
    set id [array names Lookups ${prefix}*.$dec]
    if {[string match $id ""]} { set id UNKNOWN.-1 }
    append Telnet(comment-temp) " $Lookups($id)"
}
```

The lookup table has indices in the form mnemonic.decimalValue, and could be constructed with code like this:

```
set Lookup(IAC.255) {interpret as command:}
set Lookup(DONT.254) {you are not to use option}
set Lookup(DO.254) {please, you use option}
...
```

However, the telnet include file (telnet.h) already has a nice set of comments that would be useful for this application. Rather than making up my own messages, I decided to write a little script to read the include file, find the lines in the form

```
#define mnemonic value /* Comment */
```

and use them to generate the lookup table to convert the binary values to a human-readable text. This uses the regexp command's support for sorting patterns into separate variables to recognize the lines with valid information and build the array index and value.

```
##############################################################
# proc readIncludeFile {arrayName fileName}–
#    Read an include file, and make a lookup table for all the
#    lines that match a
#    #define xxx val  /* comment */
#    or
#    #define xxx val
#    format
#
# Arguments
#   arrayName: The name of an array in the calling procedures scope.
#   fileName: The full path for the include file to be read.
#
# Results
#   Adds new indices to the arrayName in the form:
#   set arrayName(mnemonic.value) "comment"
#
proc readIncludeFile {arrayName fileName} {
      upvar $arrayName localArray
      set if [open $fileName r]
      while {[set len [gets $if line]] = 0} {
      # Use the Advanced Regular Expression "\s"
      # Class Shorthand to encode for whitespace.
      set m [regexp \
          {#define[\s]+([^\s]+)[\s]+([^\s]+)[\s]+/\*([^*]+) \*/} \
              $line all mnemonic num comment]
      # If no match, it might be a line with no comment
      if {!$m} {
          set m [regexp {#define\s+([^\s]+)\s+([^\s]+)} \
              $line a mnemonic num]
      set comment "No Comment"
      }
      # If still no match, this line must note be a #define line
      #  skip it.
      if {!$m} {continue}
      # Holler if there's a collision.  I guess a human will
      #  need to deal with this file.
      if {[info exists localArray($mnemonic.$num)]} {
```

```
            puts "COLLISION: $num"
            exit
        }
        # Everyone's happy, assign the value
        set localArray($mnemonic.$num) $comment
    }
}
```

The readIncludeFile requires Tcl 8.1 or newer to use the [\s] convention for describing whitespace.

This procedure is invoked as

```
readIncludeFile Lookups /usr/include/arpa/telnet.h
```

That covers all the pieces in this application. When you run it, it can send some simple responses to the server, and will report all the IAC messages the server sent to the client. This can be useful if you are trying to figure out what optional protocols your server supports, or if you are writing a robot that needs to talk with a telnet server.

If you run the program with the telnet server distributed on RedHat 6.2 Linux, it will return:

```
COMMENT:    please, you use option:     terminal type
COMMENT:    please, you use option:     terminal speed
COMMENT:    please, you use option:     X Display Location
COMMENT:    please, you use option:     New – Environment variables
```

If you use the program to examine a BSD system it will return:

```
COMMENT:    please, you use option:     Unknown Option
COMMENT:    I will use option:          Encryption option
COMMENT:    please, you use option:     terminal type
COMMENT:    please, you use option:     terminal speed
COMMENT:    please, you use option:     X Display Location
COMMENT:    please, you use option:     New – Environment variables
COMMENT:    please, you use option:     Old – Environment variables
```

As usual, this code for this example is available at *http://noucorp.com*.

flexible array members and designators in C9X

We've started looking at new features in C9X, the recent standards update to C. In this column we'll look at a couple of C9X features that you can use to declare and initialize structures.

by Glen McCluskey

Glen McCluskey is a consultant with 20 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

Flexible Array Members

If you've been around C for a long time, you might have encountered the following sort of usage, which goes by the name "struct hack":

```
#include <stdio.h>
#include <stdlib.h>

struct A {
    int a;
    int b[1];
};

int main()
{
    const int N = 10;
    int i;

    struct A* p = malloc(sizeof(struct A) +
        sizeof(int) * (N - 1));

    p->a = 100;
    for (i = 0; i < N; i++)
        p->b[i] = i;

    printf("%d %d %d\n", p->a, p->b[0], p->b[N-1]);

    return 0;
}
```

The idea here is that you have a struct, and one of the struct members is an array, and the array is of variable length. So you put it at the end of the struct, and you allocate extra space for the struct. So the memory for the struct object has some extra space tacked on to it, for the elements of the array.

This usage represents undefined behavior, even though it will work much of the time. With C9X, however, the usage has been legitimized, and goes by the name "flexible array member." Such a member must be the last in the struct, and the length is omitted within the []. The corresponding C9X version of the example above would be:

```
#include <stdio.h>
#include <stdlib.h>

struct A {
    int a;
    int b[];
};

int main() {
    const int N = 10;
    int i;

    struct A* p = malloc(sizeof(struct A) +
        sizeof(int) * N);

    p->a = 100;
```

```
    for (i = 0; i < N; i++)
        p->b[i] = i;
    printf("%d %d %d\n", p->a, p->b[0], p->b[N-1]);
    return 0;
}
```

No actual array space is allocated within the struct object, so we allocate extra space for N elements instead of N-1. However, the size of the object takes into account the alignment restrictions on the array. For example, with this code:

```
#include <stdio.h>
struct A {
    char a;
};
struct B {
    char a;
    int b[];
};
int main() {
    printf("%d\n", sizeof(struct A));
    printf("%d\n", sizeof(struct B));
    return 0;
}
```

the size of A will typically be 1, and the size of B will be 4.

There are some restrictions on the use of flexible array members. For example, you cannot have an array of objects, if the object type includes a flexible array member:

```
struct A {
    int a;
    int b[];
};
struct A data[10];
```

Nor can an object of such a type be used as a member in the middle of another type:

```
struct A {
    int a;
    int b[];
};
struct B {
    struct A a;
    double b;
};
```

In both cases, the flexible array member causes problems with object layout, because its total size is not known.

Designators

Suppose that you're initializing a struct object, and your code looks like this:

```
#include <stdio.h>
struct A {
    int x;
    int y;
};
```

```
struct B {
    struct A a;
    double b;
};
struct B obj = {37, 47, 12.34};
int main() {
    printf("%d %d %g\n", obj.a.x, obj.a.y, obj.b);
    return 0;
}
```

Even in this simple example, the initialization of the B object is a little confusing. The first two values, 37 and 47, get assigned to the A sub-object within B, and then the 12.34 value gets assigned to the second field of B.

Designators can be used to make such initialization much more clear. Here's an equivalent program to the one above:

```
#include <stdio.h>
struct A {
    int x;
    int y;
};
struct B {
    struct A a;
    double b;
};
struct B obj = {
    .a = {
        .x = 37,
        .y = 47
    },
    .b = 12.34
};
int main() {
    printf("%d %d %g\n", obj.a.x, obj.a.y, obj.b);
    return 0;
}
```

The notation:

```
.x = value
```

is used to initialize a specific field; the fields need not be given in order, and you can omit fields.

You can also use designators to specify particular elements in array initialization, like this:

```
#include <stdio.h>
struct A {
    int x;
    int y;
};
struct A data[] = {
    [100].x = 37, [100].y = 47
};
```

```
int main()
{
    printf("%d %d\n", data[0].x, data[0].y);
    printf("%d %d\n", data[100].x, data[100].y);
    return 0;
}
```

In this example, the designator notation is not only easier to read, but it allows you to jump ahead to initialize a specific array element, without having to specify zeros for all the previous elements. Another example of initializing sparse arrays in this way looks like this:

```
#include <stdio.h>
#define N 10000
int data[] = {[0] = 1, [N/2] = 2, [N-1] = 3};
int main() {
    printf("%d %d %d\n", data[0], data[N/2], data[N-1]);
    return 0;
}
```

You can also use designators in union initialization. The existing C rules say that an initializer for a union is applied to the first member of the union. For example, in this code:

```
#include <stdio.h>
union U {
    char a;
    double b;
};
union U obj = {12.34};
int main() {
    printf("%g\n", obj.b);
    return 0;
}
```

the double initializer value is applied to the char member, and you will get garbage when you later try to access the double value. With designators, however, this problem is easily solved:

```
#include <stdio.h>
union U {
    char a;
    double b;
};
union U obj = {.b = 12.34};
int main()
{
    printf("%g\n", obj.b);
    return 0;
}
```

Using a designator, you can initialize any member of a union.

You can use designators and flexible array members in your programming to make your job easier, and your code easier to read and maintain.

musings

When you get to read this, it will be in the depths of summer. But, today it is May Day, and the Chinese have proved to be a disappointment.

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V*.

<rik@spirit.com>

You might recall the ruckus stirred up when the US refused to say "we're sorry" about the death of a Chinese fighter pilot who collided with a US spy plane 70 miles off the coast of the Chinese mainland in April. As an afterthought, some Chinese hackers threatened a coordinated attack against the US between May 1 and May 7. Both days are important: May 1 is International Workers' Day, and May 7 is when the US sent three cruise missiles into the Chinese embassy in Serbia (oops). At least that time, the US said "sorry" right away.

I had thought that perhaps something would really happen. After all, the "1i0n worm," a worm that was (and perhaps still is) exploiting Linux systems sends /etc/shadow off to an address at china.com (which is registered in the Asia Pacific, at least), and was written by a Chinese hacker group of the same name. The Lion worm exploits BIND 8.2 (not again!!), but only on Linux systems running on x86. A portion of its install includes the setup of more automatic scanning, the t0rn rootkit, as well as DDoS agents, like trinoo and TFN2K.

If the Chinese were really using a worm to exploit and collect Linux systems worldwide, and these systems now had DDoS agents installed, they might have "systems to burn" and create a really interesting set of floods in the US. But, nothing has happened (so far). Also, the tools installed, especially trinoo, are really primitive compared with later DDoS agents, which include the ability to update themselves and execute any command as root.

More FUD

The other interesting event of the day was the issuance of CERT advisory CA-01-09 (*http://www.cert.org/advisories/CA-2001-09.html*). For one thing, this was more of a paper than an advisory. I wrote to a friend who works for CERT and suggested that they add an "Executive Summary," so that people will have at least a clue of what it is about and how important it is.

And really, I wondered, how important can it be? CA-01-09 explains some issues in how Initial Sequence Numbers (ISN) are generated, and how studies by Tim Newsham (*http://www.guardent.com/comp_news_tcp.html*) and Michal Zalewski (*http://razor.bindview.com/publish/papers/tcpseq.html*) show that the methods used by many vendors are insufficient to guard against even a "weak" attack that relied on guessing the ISN. I have no argument with the analyses, which date back to seminal papers by Robert Morris (1985, *ftp://research.att.com/dist/internet_security/117.ps.Z*) and Steve Bellovin (1989, *http://www.research.att.com/~smb/papers/ipext.ps*). How could I argue against these guys?

But what I was wondering about is just how relevant this issue is to current configurations of various servers' OSes. What got people's attention in 1994 was the very interesting attack against Tsutomu Shimomura's home network on Christmas day, the very attack that led to the search for Kevin Mitnick (who in all likelihood did not create the attack and might not even have launched it). The attack actually turned the warnings of Morris and Bellovin into something real for the first time. You can find a variant of this attack named rbone2 on some exploit sites even today.

The attack relies on finding a pair of systems that have a trust relationship and rsh servers that are reachable by the attacker. Now, anyone with any security training should

be aware that using rsh is a very bad idea. This lapse led me to believe that Shimomura could not have been a security guru since he was using rsh. What Shimomura had done was use TCP wrappers to provide access control based on the IP source address, and then allowed root access between his pair of trusted Sun systems. That is what the attacker exploited using a tool like rbone2.

The tool first uses a SYN flood to "disable" a port on the trusted server. Before this attack, a SYN flood was something that happened to busy Web servers (also a new thing in 1994). Most servers had a connection queue for each port that was only about ten entries deep. If an attacker sent TCP SYN packets with spoofed source addresses, the server's TCP stack would send SYN-ACKs, then wait for responses that would never come, eventually timing out after 75 seconds. So, all an attacker had to do to SYN flood a service was to send enough packets to keep the connection queue filled. The attack tool sent 20 packets.

Next, the tool sent more SYN packets, but this time to the trusting host, and using the tool's real IP address. This allowed the tool to collect ISNs from the soon-to-be victim. Before UNIX vendors bothered to generate pseudo-random ISNs, ISNs were VERY predictable, with each subsequent ISN being 128,000 greater than the previous (unless more than one second had passed, which would also increment the value by 128,000). The attack tool collected a sequence of ISNs, resetting the connection each time instead of letting it complete (which prevents TCP wrappers from ever being invoked and logging the failed connection).

Now, the attack tool has a potential value for the next ISN and can proceed. The attack tool spoofs the IP address of the trusted server and sends a SYN packet to the victim. The victim responds to the trusted server, but the port used is the one that was SYN flooded, and the packet is discarded. Then, the attack tool sends a spoofed reply, which must include the victim's ISN (plus one) in the response if it is to succeed. If the attack tool has guessed the victim's ISN correctly, it can finish the attack.

Keeping in mind that the attacker never sees any response packets, what can be done to the victim? The attack tool sends packets to execute commands as root to open a hole in the victim's defenses, allowing remote access as root (for example, `echo ++ >> /.rhosts` is part of this attack). The only way the attacker can tell if the attack has succeeded is to try an rsh to the victim, and if this works, the attack has succeeded. If not, well why not try again until it does work?

My own problem with this is twofold. How many people would consider using the r commands and trust relationships today? SSH provides a drop-in replacement for the r commands that includes digital signatures that positively identify both the client and the server to each other. So spoofing the source IP address won't work anymore. But perhaps people are still using the r commands.

The advisory also mentions TCP hijacking, but this requires more than guessing the ISN. You must also know the socket information, presumably by sniffing, in which case you don't need to guess the ISN.

The CERT advisory points out that even if the ISN gets incremented with pseudo-random numbers, it is still possible to guess the increment, given enough attempts. The advisory points out that given a series of ISNs, they will tend to fall around an "average" value over time, and given enough attempts, the attack tool could still guess an ISN. Sure. Why not?

What is easier to do, and more likely, is to guess an ISN and send RESETs to break existing connections. In the case of shutting down an existing connection, you still need the socket information. If the attacker has that, the ISN (actually the acknowledgment value) only needs to be within the TCP window in order to be accepted, and then the RESET flag will cause the connection to be terminated. If you are curious about typical TCP Window values (used by the receiver to control how many bytes of data the sender may transmit without receiving an acknowledgment), check out the nmap-os-fingerprints file (get nmap from *www.insecure.org*), and you can see that this value varies from as small as 512 bytes to as much as 32K-1 (Windows NT/SP3, W=7FFF).

But this is still a problem. How does the attacker know the socket information (source port, source IP address, destination port, destination IP address) without sniffing packets? The destination info is easy, as you can port scan the destination server and glean that info. But the source info is much more difficult, as (at the very least) the client port will be some value between 1 and 65535.

There are cases where the client port is easy to guess. Communications between DNS servers may use the same port at each end (53/tcp). Is this the thing that CERT is so worried about? Maybe. A determined attacker could disrupt updates to root servers, with the effect that new DNS info could not be received. Remember what happened when Microsoft's DNS servers could not be reached in January? All of their domains, and related Web and email servers, effectively disappeared from the Internet.

Well, sorry, but this still seems pretty far-fetched to me. I also wondered about BGP4, used to exchanged routes between core routers. BGP4 requires that peer routers keep a live TCP connection at all times, so breaking a connection (and keeping it down for some time) would force the peers to announce new routes. The problem with ISN guessing affecting routers using BGP4 is that they use RFC2385, and sign their TCP headers, preventing spoofing. To be honest, I found this interesting, as someone decided that spoofing was enough of a threat to BGP4 that RFC2385 was considered that solution (*http://www.faqs.org/rfcs/rfc2385.html*, published in 1998).

So, should you worry about the ability of attackers to guess ISNs? If you are using the r commands, and have no firewall to protect the servers running these commands, yes. If you are worried about people hijacking or resetting TCP connections, no. The true solution is to use encrypted connections, such as SSH, SSL, or IPSec. While improving the methods used for ISN generation is important, encryption is the better countermeasure (as the CERT advisory notes).

Also, congratulations to OpenBSD and the Linux developers for having already implemented strong ISN, a la RFC1948. Note that FreeBSD has included the OpenBSD solution, and that other OS vendors, like Sun, support more secure ISN generation as a tunable kernel parameter, but not by default.

MS Horror Story

I just spent the last couple of hours installing Windows 2000 Professional. I guess Microsoft thought that if you got the Pro version, it should be difficult to install (for pros only, get it?). The fifth reboot just completed, soon after I elected to "Finish configuration of this server later." Well, I was sure that was what I selected, but Win2K has a "mind" of its own, and went about setting up Active Directory and other services after I told it "later."

I have seen a preview of the SAGE 2000 Salary Survey, and it seems that lots of you have to manage Windows servers (second only to Solaris). Sorry to hear that. The other inter-

I have seen a preview of the SAGE 2000 Salary Survey, and it seems that lots of you have to manage Windows servers (second only to Solaris). Sorry to hear that.

esting tidbit was that people who only manage Windows servers get paid less (a LOT less) than people who also manage UNIX systems. I really wondered about that. Is this merely because people consider UNIX so much more difficult than Windows, or because basic Windows administration is based on carrying about a CD pack, and reinstalling software frequently? Sorry, didn't mean to criticize anyone, but I did quote a UNIX sysadmin in a past column who described Windows sysadmin as exactly that. At least it would explain the salary gap.

Ah, Win2K is finally rebooted. After installing the (!#$!!) software that I need that will only run under Windows (yes, I know about and have installed StarOffice, and yes, I still use VMWare on my notebook, thanks), I have to reboot again. I notice that Win2K has a nifty new feature where your menu choice fades out after you have selected it. I call this the "LCD screen simulation effect," so that you can get some of the feeling of having an LCD screen without spending the money.

Before I conclude this column, I would like to thank the three Libertarians who were so annoyed by my February column that they wrote me letters many times longer than the actual column itself. I do read my email and will respond to any letters you care to send me, and am still wondering exactly why my cynicism only annoyed Libertarians.

ISPadmin

Usenet News

by Robert Haskins

Robert Haskins is currently employed by WorldNET, an ISP based in Norwood, MA. After many years of saying he wouldn't work for a telephone company, he is now affiliated with one.

<rhaskins@usenix.org>

Introduction

In this installment, I look at many diehard Internet users' favorite application and every service provider's headache: news. Usenet news is defined by RFC977 (NNTP proposed standard) and RFC1036 (Usenet message standard).

The problem of news is a difficult one for a service provider, due to the following attributes:

- Very high volume of both posts and news data itself
- Very high (exponential?) rate of increase in both number of posts and MB of data year after year
- High number of end users
- The distributed distribution model implemented by NNTP, making reliability and "correctness" (i.e., posts containing all of the data they are supposed to) problematic
- Infrastructure costs, including bandwidth and disk space
- Personnel costs to monitor spam and illegal articles originating on their network, as well as to manage the news infrastructure itself
- Legal problems associated with spam, illegal pornography, and warez (illegal software)

Small Provider Infrastructure

A small provider (and most enterprises who offer news to their employees for that matter) will likely utilize a single machine for news. This single machine performs all three functions typically required in a news infrastructure: inbound news relay, outbound news relay, and serving news to clients.

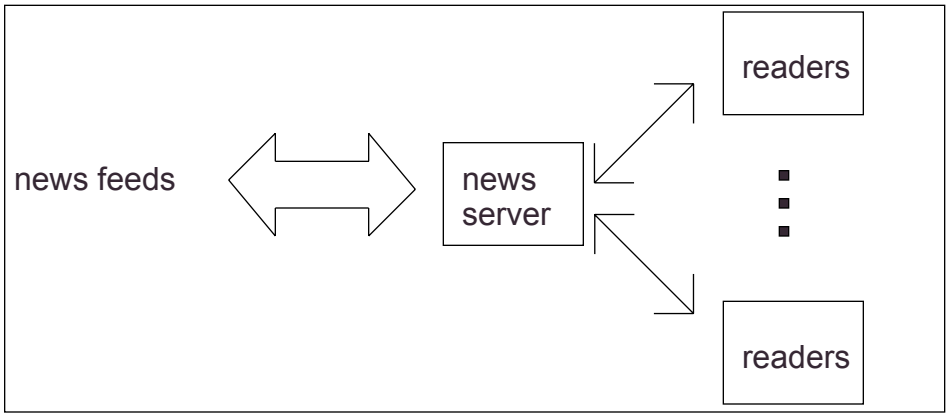


Figure 1

Figure 1 contains a diagram of how a small provider might set up their news infrastructure. The center box labeled "news server" handles all three basic news functions: inbound news relay, outbound news relay, and client news readers. The inbound articles come into the machine from the various sources of news (peers, commercial providers, upstream ISPs, etc.). The outbound articles leave the server though the outbound news connection(s) (usually the same sources as the inbound news streams). All of the customers who want news point their news clients to this same machine.

It is very likely that a small provider is not going to want to deal with the hassles of news and will outsource news to a provider like Critical Path (formerly Supernews/RemarQ). (The References section contains a pointer to a listing of news providers at the Open Directory project home page.) While some larger providers do utilize commercial news services providers, it is usually more cost effective for a big provider to set up their own news infrastructure.

Large Provider Infrastructure

A large provider is likely to deploy separate machines (or groups of machines) for each function: inbound news relay, outbound news relay and client news serving. Of course, functions can be combined; for example, inbound and outbound news relay can be the same machine if the inbound and outbound news volume isn't too high.

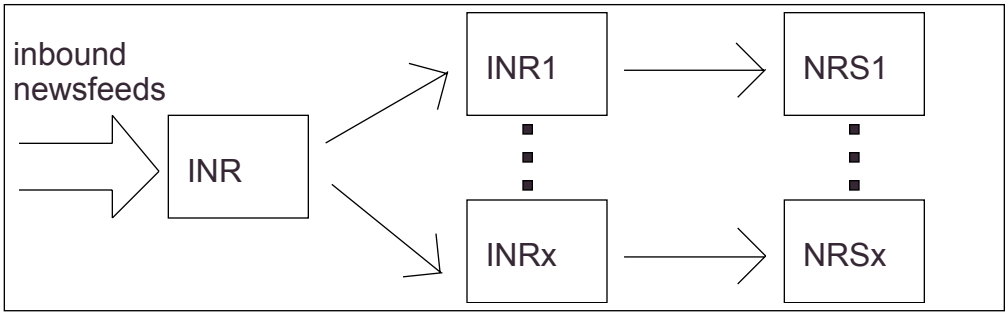


Figure 2

Figure 2 outlines how a larger provider might set up their inbound news infrastructure. The box marked "INR" illustrates an inbound news relay machine. This machine takes all off-site incoming feeds and consolidates them into a single feed. Only the very largest providers would need more than one INR machine for performance reasons. (Of course, they may have multiple INR machines for redundancy purposes.) A single machine to consolidate incoming feeds keeps transit costs to a minimum.

Multiple inbound news relay machines accept feeds from the inbound news relay machine(s). Each inbound news relay machine sends news articles to multiple news reading servers (indicated by "NRS1" and "NRSx") which news reading clients (not shown) attach to. Note that depending upon the news server hardware and software running on the news reading servers, each server can feed hundreds of news clients concurrently. The first class of machines to require scaling is usually the news reading servers, followed by the inbound news relay.

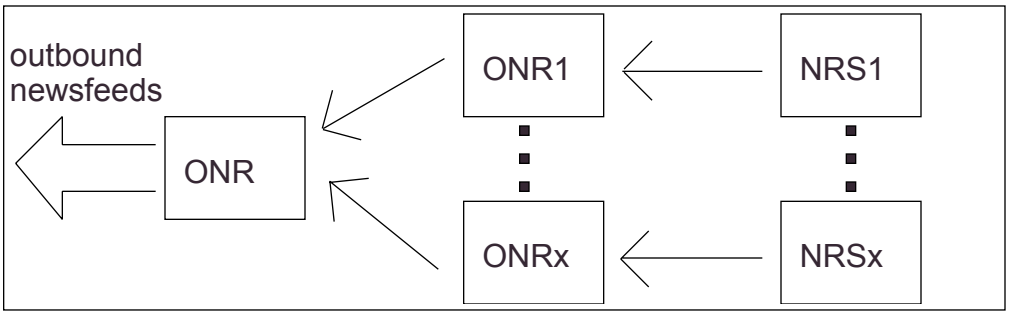


Figure 3

Figure 3 shows how a big provider could set up their outbound news (posts originating on their network) infrastructure. The boxes marked "ONR1" and "ONRx" indicate outbound news relay machines, which take articles from the news reading servers (labeled "NRS1" and "NRSx") and send them to the machine labeled "ONR". The outbound news relay might be located on the same machine that provides the inbound news relay function, depending upon the number of articles originating on the providers network. This outbound news relay machine is tasked with sending articles originating on the providers network to the Internet at large through the outbound news feeds previously configured. The outbound news relay machines (machines labeled "ONRx") are typically the last part of the news infrastructure that requires scaling, since news clients don't usually originate many news articles.

Cyclical News File System (aka The Trash Can)

The vast majority of news implementations utilize either Internet Software Consortium's INN (originally written by Rich Salz) which is open source or Openwave's Typhoon/Cyclone series of commercial software. Before discussing the applications in particular, a short history and discussion of the circular news file system would be helpful.

Prior to the implementation of the cyclical news file system (CNFS) within INN, many providers (including Time Warner Cable of Maine and Ziplink) who had implemented INN 1.x switched to Typhoon and Cyclone because INN simply could not handle the load, or expire articles automatically. Typhoon/Cyclone (and, more recently, INN) both feature an implementation of a cyclical news file system that eliminates many of the headaches when managing a news infrastructure, in particular, article expiration.

Article expiration is the process by which articles are "cancelled" and deleted from the list of available articles for download. As articles "age," they are expired. Historically, article expiration was handled by setting parameters within an INN configuration file. Every day at a certain time, a process ran which deleted articles that met the criteria set in the configuration file, and all of the news indexes were re-indexed. This process could take hours for a large news system, frequently causing service interruptions.

With the rapid growth of the size of newsfeeds, the partitions articles were stored on frequently filled up if an administrator was not diligent in keeping the expiration configuration file up-to-date with the added news groups. Also, performing the expiration would often cause service interruptions due to the load put on the news server while running the article expiration.

A cyclical news file system has no concept of article expiration. Once the disk is filled, the oldest articles are simply overwritten with new articles in a cyclical news file system. Therefore, there is no need to perform the CPU-consuming expiration process and its associated overhead. The INN 2.0 release includes CNFS support.

Openwave's Twister and Cyclone

Twister and Cyclone are commercial service provider-grade news server implementations. Many service providers utilize these products for serving news to their customers. Some of the features of Openwave's Twister and Cyclone products include:

- Virtual server support
- Customizable anti-spam filtering
- Synchronized article numbering across the entire news server infrastructure
- Real-time statistics and logs capable of generating bills
- Post filtering
- Automated moderator support
- Feeds automatically adjusted without administrator intervention for optimal throughput and efficiency.

Openwave has a free version of their discussion software named Breeze. Of course, there are limits to how many feeds and readers can connect to it, but it might be worth some investigation if you are in the market for Usenet news server software.

Internet Software Consortium's INN

INN is freeware and doesn't have all of the bells and whistles that commercial applications like Openwave's servers have. However, it is a fully functional news server and perfectly capable of serving news. The advent of the CNFS in INN makes it much more robust and usable in a service provider environment. Features of INN 2.3.1 include:

- Python, Perl, and Tcl authentication and filtering plug-in support
- News reading over SSL
- Email gateway to news
- Exponential backoff for posting, enabling some level of anti-spam support

For many providers, large and small, INN is a fine solution to the problem of Usenet news. If the added features of a commercial news product (like article synchronization, virtual server support, and real-time statistics) are required, then a provider would likely utilize a commercial-grade server.

News Client Software

News client software bears a brief mention. Both Microsoft and Netscape browsers contain client news reading capability. While they both can read news, I personally find them not nearly as functional as the Forte Agent news-reading client. For those folks who have been around since before the GUI days, you can still read news from the UNIX command line utilizing tin, trn, pine or a multitude of other character-based news readers. If you are interested in finding out more about news clients, please check out the appropriate Web site in the References.

Storage Considerations

When designing news infrastructure, many details must be considered. In the area of storage, single-disk spindles (i.e., not RAID or other fault-tolerant storage technology) are usually utilized for storage as losing articles is a tolerable event. Also, backups are almost never performed (except for those news providers who archive such things) because once again, losing articles is acceptable. Once the hardware failure is repaired, news will begin filling the disks again very rapidly!

News articles can be stored and shared via NFS mounts. Historically, many problems arose, including file locking and performance issues, from using NFS for article storage, which accounts for the limited use of NFS in news implementations.

It is not recommended that NFS be utilized for news implementation; storage area networks, or SANs, provide a much better way to achieve similar functionality and performance.

Other Considerations

As you are probably aware, Usenet news hosts thousands of news groups in a multitude of languages. For providers with networks located solely in the US, it is sufficient to carry the 50,000 or so English-only groups. International service providers would likely carry a complete feed with all non-English groups as well.

End subscribers control the groups to which they subscribe. When news clients initially connect to news servers, the servers will query clients as to what group headers to download (usually all are downloaded). Once the group headers are downloaded, end users can subscribe (download article headers within each group) to whatever set of groups interests them, and then download individual article bodies that they wish to view.

Most providers carry local news groups (a group dedicated to restaurant reviews in the provider's city, for example). In fact, the Openwave series of news servers enables "virtual groups" to be located across servers and only visible to certain classes of clients (for example, the customers of a particular ISP in a wholesale ISP's news infrastructure). This is a very useful feature for a wholesale service provider.

One might wonder how much effort and hardware it took to run a news infrastructure at a moderate-sized ISP. At Ziplink, we had a moderate-sized English language only news infrastructure containing 50,000 groups and 200 news clients. These clients were served by two Sun Ultra 5 machines, each with 512MB of RAM and approximately 36GBs of disk space. One machine ran Cyclone and was the inbound and outbound

news relay, while the other machine ran Typhoon and was the news reader machine. The load on either machine was never higher than 1, and usually between 0.3 and 0.5. The aggregate feeds were on the order of 2 Mbps, from a handful of UUNET news feeds and several news peers.

Occasionally a news spam complaint will arrive in the abuse mailbox of a provider. Usually, it is very easy to track down the perpetrators of news spam, as the logs and message headers themselves contain exactly when and where the message originated. Forging message headers makes this process much more difficult, but the logs again make it easy to determine positively whether or not a message originated on a particular provider's network. Generally, Usenet news spam is much less of a problem for an ISP than junk email. In 2.5 years at Ziplink, I handled one Usenet spam complaint but hundreds of unsolicited commercial email complaints.

Legal Aspects of News

I've asked John Nicholson (the lawyer who writes in *;login:* about legal issues surrounding computers) to cover ISP legal areas, as I'm not an attorney. Usenet news would definitely be an important topic for any discussion around service provider legal liability.

Most ISPs consider themselves "common carriers." Having "common carrier" legal status would exempt ISPs from liability of what is carried over their infrastructure. How true this belief really is, I am not sure. If considered a "common carrier," a service provider cannot be held liable for pornography or illegal software originating or residing on their infrastructure. (An example of an entity with common carrier status would be the US Postal Service; the USPS cannot be held responsible for someone sending illegal drugs through postal mail.)

Most if not all providers perform no *content based* censoring (moderating) of what content flows through their network. Of course, decisions *not* based on content but strictly on technical capacities and related areas (for example, limits placed on news articles based on the amount of disk space or network bandwidth available) is an acceptable means of controlling one's destiny as an ISP while not jeopardizing the potential for "common carrier" legal status.

Conclusion

Providing Usenet news functionality can be a difficult task for any provider. For a smaller provider, one machine can handle inbound, outbound, and news reading capability. For a larger provider, functionality is split up based inbound, outbound, and news reading capability.

Openwave's discussion products function as a good commercial news server, while INN remains the open source stalwart. Spam is not too much of an issue when it comes to news, and those who do spam are relatively easy to catch. Most ISPs do not filter groups for fear of losing "carrier" status. At this point it is unclear ISPs have this status under any circumstances.

Next time, I'll take a look at how service providers deploy their name service infrastructure. In the meantime, please send your questions and comments regarding ISPs, system administration, or related topics to me.

REFERENCES

NNTP proposed standard (RFC977): *ftp://ftp.isi.edu/in-notes/rfc977.txt*

Usenet message standard (RFC1036): *ftp://ftp.isi.edu/in-notes/rfc1036.txt*

RFC Editor: *http://www.rfc-editor.org/*

Critical Path Supernews/RemarQ: *http://www.supernews.net/*

Open Directory Project list of commercial news providers: *http://dmoz.org/Computers/Usenet/Feed_Services/*

ISC's INN: *http://www.isc.org/products/INN/*

Openwave's discussion server software: *http://discussion.openwave.com/*

Microsoft Internet Explorer: *http://www.microsoft.com/windows/ie/*

Netscape Navigator: *http://home.netscape.com/browsers/index.html*

Forte Inc., Agent / Free Agent: *http://www.forteinc.com/*

tin: *http://www.tin.org/*

pine: *http://www.washington.edu/pine/*

trn: *http://trn.sourceforge.net/*

what are your intentions?

by Brent Chapman

Brent Chapman is an IT infrastructure and network architecture consultant based in Silicon Valley. He is coauthor of the O'Reilly & Associates book *Building Internet Firewalls*, and creator of the Majordomo mailing list management package. He is also a founding member of SAGE and was the original SAGE postmaster.

brent@greatcircle.com

Air traffic controllers, with their ability to use radar to see exactly what aircraft are doing, seem to be the very picture of omniscience in their domain (at least until the 30-year-old mainframe in the basement blows another vacuum tube, and the whole screen goes blank). However, there's a big difference between "all-seeing" and "all-knowing." A controller might be able to see exactly what a given aircraft is doing, but be totally in the dark about *why* the aircraft is doing that. If the controller doesn't know and can't figure it out, the pilot of the aircraft in question soon hears something like "Cessna Four Three Zero Papa, what are your intentions?" The controller needs to know what each of the pilots in their care is trying to accomplish, so that the controller can coordinate all their actions to get them each safely and efficiently to their destinations.

Whether we realize it or not, as IT professionals we often face similar situations. Any competent system administrator can examine a system and determine the details of *how* the system is configured, but they may still be totally in the dark about *why* it is configured that way. Without understanding the "why" behind a system's configuration, it's much more difficult to make changes or updates to the system successfully, because it's much more difficult to predict the effects (positive or negative) that those changes will have. It's also difficult to evaluate how well a system is currently doing its job, when you're not entirely sure just what job it was designed to do.

When documenting our systems and networks, we often spend a huge amount of effort documenting what they are, how they're configured, and how to use them, but little or no effort documenting why we set them up that way. While a certain amount of "what" and "how" documentation is definitely called for, particularly as a map of the situation, there are two problems with producing only this form of documentation. First, much of this documentation quickly gets out of date, particularly if it covers system-level details like how much memory and disk space a system has; when you need to know such info, it's usually better to get it from the system directly. Second, even if it's kept accurate, this documentation doesn't tell a later reader (months or years after the system was deployed) anything at all about why the system is configured that way, which makes it more difficult to repair, extend, replace, or retire.

Examples of questions that are often left unanswered in the documentation include:

- Why are we using a given package or version for a particular service?
  - Why Postfix? (rather than Sendmail or QMail or whatever)
  - Why Oracle? (rather than MySQL or Postgres or whatever)
  - Why BINDv8? (rather than BINDv9)
  - Why RedHat? (rather than Debian or SuSE)
- Why are we using a particular vendor or service provider?
  - Why Dell? (rather than Compaq or Micron or whoever)
  - Why Sun? (rather than HP or SGI or whoever)
  - Why UUNET? (rather than XO or PSINet or whoever)
  - Why Exodus? (rather than Level3 or Digital Island or whoever)
- Why are certain services provided by particular systems, rather than some other system?
- Why are certain services grouped onto particular systems?
  - Why are other services given their own systems?
- Why are our standard systems what they are?

- Why this config for an engineering desktop?
    - Why this config for an engineering laptop?
    - Why this config for a non-engineering desktop?
    - Why this config for a non-engineering laptop?
- Why is our IP address plan set up the way it is?
    - Why do we allocate a /24 to each office?
    - Why do we skip every other /24 in the allocation?
    - Why do we allocate all the point-to-point /30s as we do?

Competent IT professionals usually have good reasons for the things they do. They can usually tell you those reasons, if you ask. Months or years later, however, it can be difficult to obtain this information; the professionals in question may have worked on so many things since then that they no longer remember, or they might not even be with the organization anymore. A much better approach is to create a short explanation (maybe just a paragraph or two) of why a system is the way it is at the time the system is configured.

These reasons are often discussed and debated among staff while the system is being designed and deployed; all you need to do is capture the conclusions of those discussions and debates. At a small shop, an archive of your internal IT email discussion list might be sufficient, if you've got a good search mechanism. Or you might want to go one small step further and establish a special archived email alias just for these "wisdom" summaries; the trick will be to remember to create such summaries and send them to the alias for recording.

Growth plans are a related issue. Competent IT professionals usually include avenues for efficient future growth in their designs, but those avenues often aren't obvious later, especially if the person looking for them wasn't a participant in the original design process. These growth plans and thoughts should be documented when the system is designed and deployed, so that they can be used when needed in the future.

Assumptions about the environment and its future are another thing that should be documented when a system is designed and deployed. Documenting these assumptions will help later IT staff determine when a design is no longer suitable for the current (ever changing) situation, and should be revised or replaced. For example, if you design an IP address plan to accommodate a dozen large field offices, total, over the next five years, but your company has shifted plans and is instead now opening a dozen small offices per month, you probably need to rethink your address plan.

Assumptions can also take the form of constraints, which might not apply in the future. If you don't document them at design and deployment time, however, it will be more difficult to take advantage of later changes that ease those constraints. For example, you've probably heard the joke about the mother teaching her child their family's traditional method of preparing a roast, which starts with cutting the roast in half. The child asks "Why?" and the mother says, "Well, that's the way we've always done it; that's the way my mother taught me to do it." The child, being a naturally curious sort, goes and asks Grandma "why cut the roast in half?" and gets the same answer. The child recurses through ancestresses until eventually reaching one who answers, "Because it wouldn't fit in the pan that I had when I was a young bride." That's a perfect example of a system carried forward far longer than necessary, because nobody realized that the relevant constraint (the size of the pan) no longer applied.

To get a good idea of what your "why" documentation should cover, imagine the questions that a new hire would ask if they were made responsible for that system or service.

Assumptions about the environment and its future are another thing that should be documented when a system is designed and deployed.

As a consultant, asking "why?" is often the most important service that I provide to my clients.

Assume that this person is a competent system administrator, and knows how to find and use vendor documentation for the system or service; they'd still want to know things like who the primary users are, why this particular platform and/or software package was chosen to provide the service, what the growth plans are, what the unusual or subtle aspects of this configuration are, and so forth.

As a consultant, asking "why?" is often the most important service that I provide to my clients. The act of explaining their concerns or goals to an outsider (me) often clarifies those concerns and goals. This, in turn, often makes a range of solutions clear as well, and the task becomes one of evaluating and choosing between those solutions, then implementing the chosen solution. There's no magic to hiring a high-priced consultant for this type of exercise (though I certainly don't mind the business!); you can apply this same method very well yourself, if you just take the time to do so.

So... what are your intentions?

setting up BIND8 in a change-rooted environment on solaris

by Timo Sivonen

Timo Sivonen is a consultant working for Greenwich Technology Partners. His interests include operating systems, security management and model railroads.

tljs@greenwichtech.com

Introduction

Probably the majority of all BIND installations in the world are configured to run named as super-user. This implies full access to the operating system, which probably is the most dangerous configuration possible. Any vulnerability in BIND, such as a buffer overflow bug, will put the entire underlying operating system at risk. However, there are two protection mechanisms in BIND8 to limit the consequences of a buffer overflow: running named in a change-rooted environment and changing the owner of the process to other than super-user. Although the measures in the following text will not eliminate the potential and real security vulnerabilities in BIND, the goal is to do what can be done to contain the damage caused by an attack.

Changing the root directory of a process to something other than the system root ('/') is a very effective method to limit the execution environment to the bare minimum. The rationale is to withdraw all vulnerable system tables and databases (e.g., /etc/passwd) from the reach of the service process. If the password database does not exist, it cannot be exploited. A change-rooted execution environment also permits file and directory permissions more rigorous than otherwise possible. Figure 1 shows the directory structure of our setup.

Simply moving the process to a change-rooted environment is not enough. Although only the super-user can execute the chroot() system call, the super-user can also cancel the effect of chroot() and break out from the change-rooted area. Simon Burr[1] has put together an excellent Web page on how to attack against chroot(). Consequently, we must follow the principle of least privilege and change the user owning named from root to, for example, an unprivileged bind account. Of course, there must not be any set-user-id executables in the change-rooted area.

Since BIND-8.1.2, one has been able to use a command-line option to instruct named to change-root itself immediately after it has started up, and another option to change the user-id the process is running on. Sun Microsystems has been distrib-

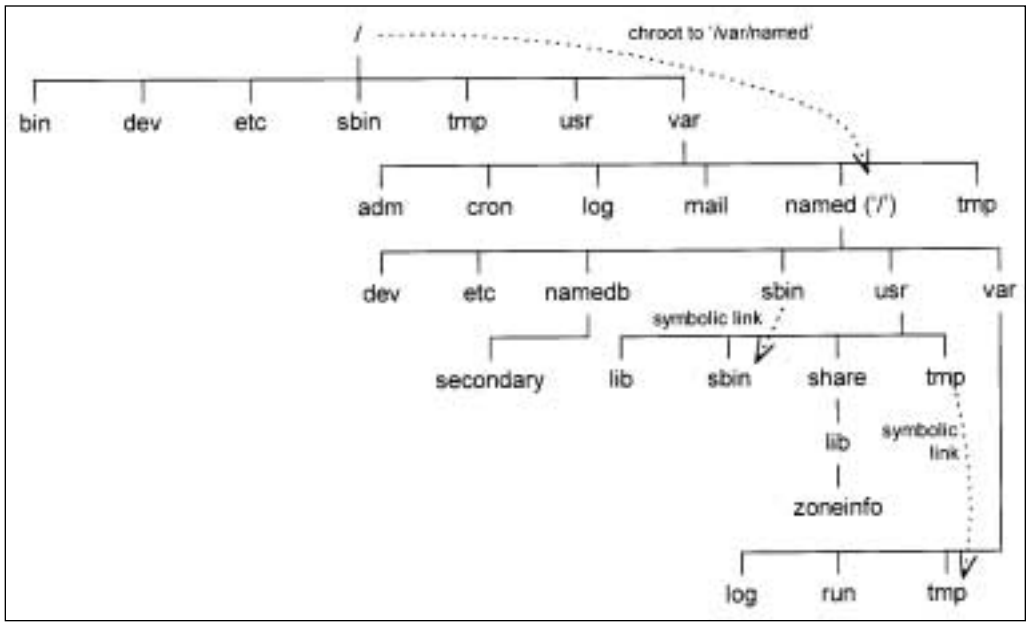


Figure 1. Directory hierarchy

uting BIND8—version 8.1.2—since Solaris 7, but sadly, Sun's version (in.named) has neither feature available. This is strikingly disturbing since change-root is never completely safe if the process is owned by root.

In the following text, we will take a generic approach and describe how to perform change-root first and then execute named in the restricted environment. This approach has two advantages:

- The process won't have any open file descriptors outside the change-rooted area.
- Since you do not rely on chroot() internal to a program, you can use the same method to change-root other unix services as well. There is also Wietse Venema's very useful chrootuid[2] utility that not only changes the root directory but also launches the process with the given user-id.

Obviously, there has been little improvement in system security if one runs BIND in a restricted environment yet the operating system has all the default network services such as telnet, FTP, XDMCP and others available to the world. SANS' Step-by-Step Guide[3] is a good source for instructions how to build a hardened Solaris system.

Getting Started

In the following text we will assume that you have either built BIND-8.2.3 from the source,[4] or you have obtained the binaries from Sunfreeware.com.[5] You can install the binaries and the configuration files to the default locations, since we will have to copy those to the change-rooted hierarchy.

The first step is to decide where to put our BIND hierarchy and create bind, the BIND account, in /etc/passwd. The BIND directory will be the home directory of the account.

It is customary to have all BIND-related files and directories under /var/named but you can also use some other location if your /var is limited in space. We will need approximately 7MB of disk space for the base setup, and you probably want to reserve some additional space for your logs, zone, and dump files.

Edit /etc/passwd, /etc/shadow, and /etc/group to set up the bind account. In the following text we will discuss the entries that must be added in each file.

```
/etc/passwd:
bind:x:65533:65533:Berkeley Internet Name Daemon:/var/named:/nonexistent
```

bind does not have to have any specific user or group id, as long as those do not conflict with any existing accounts. Since bind is meant to be a restricted account with minimum access rights and no login, it is a good practice to assign a first available id in descending order from 65535, the highest possible user-id. Note that the home directory of the account points to the root of the BIND hierarchy. This is not absolutely necessary yet it is quite handy.

```
/etc/shadow:
bind:NP:6445::::::
```

The password field for bind in /etc/shadow must contain NP to disable logins.

```
/etc/group:
bind::65533:bind
```

To be complete we also want to set up a group for bind. You should select the highest available group id for the account. The only member of the group is bind itself.

Files and Directories

Careful construction of the change-rooted directory hierarchy is critical to our setup. The guidelines described in the manual page of in.ftpd[6] offer a good starting point.

1. Create the /var/named directory.

```
# mkdir /var/named
# chown root:root /var/named
# chmod 551 /var/named
# ls -ld /var/named
dr-xr-x--x 4 root root 512 Feb 11 14:04 /var/named
```

Create the /var/named directory and set its owner and group to root. Set the directory permissions to mode 551 to permit read and directory access for the owner and group and directory-only access for everyone else (including bind).

2. Change your default directory to /var/named and create the first-level directories under named/.

Here we create the top level of our change-rooted environment. There will be only a few directories in our structure named will need write access to. For example, there has to be a read-write directory for named to store the transferred zone files.

```
# cd /var/named
# mkdir dev etc namedb usr var
# ln -s usr/sbin .
# chown root:root *
# chmod 551 *
# chmod 751 namedb
# ls -la
total 12
dr-xr-x--x      4 root   root     512    Feb 11 14:04      .
drwxr-xr-x      4 root   root     512    Feb 11 14:04      ..
dr-xr-x--x      4 root   root     512    Feb 11 14:04      dev
dr-xr-x--x      4 root   root     512    Feb 11 14:04      etc
drwxr-x--x      4 root   root     512    Feb 11 14:04      namedb
lrwxrwxrwx      1 root   root      10    Feb 11 14:04      sbin -> usr/sbin
dr-xr-x--x      4 root   root     512    Feb 11 14:04      usr
dr-xr-x--x      4 root   root     512    Feb 11 14:04      var
```

Our change-rooted sbin/ directory is a symbolic link to the change-rooted usr/sbin/. The purpose of this step is to store all executables and the corresponding shared libraries into a single hierarchy. Consequently, there will be no need to ever change the usr/ hierarchy after it has been set up. This serves the purpose of creating the change-rooted environment in the first place.

3. Change your directory to dev/ and create the necessary device nodes.

The devices are specific to your operating system and possibly even the version you are running. Therefore you should verify the major and the minor device numbers with 'ls -lL' from the system /dev directory before proceeding with creating the devices. Moreover, five out of seven devices we need are specific to Solaris: the only generic devices are dev/null and dev/zero.

```
# cd dev
# mknod conslog c 21 0
# mknod log c 21 5
# mknod null c 13 2
# mknod syscon c 0 0
# mknod tcp c 11 42
```

```
# mknod udp c 11 41
# mknod zero c 13 12
# chown root:sys *
# chown root:tty syscon
# chmod 666 conslog null tcp udp zero
# chmod 640 log
# chmod 620 syscon
# ls -la
total 4
dr-xr-x--x      2 root  root       512     Feb 11   15:06    .
dr-xr-x--x      8 root  root       512     Feb 11   14:59    ..
crw-rw-rw-      1 root  sys     21,   0     Feb 11   15:05    conslog
crw-r-----      1 root  sys     21,   5     Feb 11   15:05     log
crw-rw-rw-      1 root  sys     13,   2     Feb 11   15:05    null
crw--w----      1 root  tty      0,   0     Feb 11   15:05    syscon
crw-rw-rw-      1 root  sys     11, 42     Feb 11   15:05    tcp
crw-rw-rw-      1 root  sys     11, 41     Feb 11   15:05    udp
crw-rw-rw-      1 root  sys     13, 12     Feb 11   15:05    zero
```

We need to set up the dev/ directory for the proper operation of the network and syslog. Since BIND uses TCP and UDP, both dev/tcp and dev/udp device files must be present. These files should be readable and writable by the world (mode 666).

The files dev/conslog, dev/log, and dev/syscon control logging to the system console and syslog. BIND must be able to write to these devices for logging to work properly. One significant advantage of having these devices in the change-rooted area is the ability to collect log messages from BIND to syslog.

Traditionally the default log device, /dev/log, has been a UNIX domain socket. The Solaris /dev/log is a STREAMS device that nicely circumvents the boundaries of the change-rooted area. As a result, you don't have to specify additional log sockets for each change-rooted hierarchy: creating a private dev/log in your dev directory is enough.

Another benefit from using dev/log and syslog for logging is to move all log information beyond the reach of named as soon as a new entry is generated. This makes it considerably more difficult for an attacker to tamper with the logfiles.

The null device is there for garbage collection purposes. Any program, including BIND, may want to redirect its input or output to /dev/null, and an attempt to open a non-existent device would surely generate an error message.

The zero device is required by ld.so to set up shared libraries properly. If this device is missing you won't be able to start any change-rooted programs that utilize shared libraries.

4. Create the necessary system files in etc/.
Unlike the real /etc, our change-rooted etc/ will have only the minimum number of files, i.e., passwd, group, and netconfig. You have to change your default directory to ../etc/ to create these files.

In the following text we will use the ex line editor to create the system files. This is because it is easier to show all the right keystrokes for a line editor than for vi or textedit. Of course, you can always choose to use your favorite editor but you cannot use admintool since it can only modify the default system databases.

```
# cd ../etc
# ex passwd
```

```
"passwd" [New file]
:a
root:*:0:1:Super-User:/:/nonexistent
bind:*:65533:65533:Berkeley Internet Name Daemon:/:/nonexistent
.
:w
"passwd" [New file] 2 lines, 101 characters
:q
```

Our minimal passwd file will contain only two accounts, root and bind. These two are needed since we will start named as the super-user, but the owner of the process will change to bind as soon as possible. There will not be other user accounts.

```
# ex group
"group" [New file]
:a
root::0:root
other::1:root
sys::3:root
tty::7:root
bind::65533:bind
.
:w
"group" [New file] 4 lines, 54 characters
:q
```

Again, group will contain only the most necessary group entries. We have to include sys and tty, since we copied the ownerships and permissions of our change-rooted devices from the real /dev.

```
# ex netconfig
"netconfig" [New file]
:a
udp     tpi_clts        v  inet  udp  /dev/udp    -
tcp     tpi_cots_ord    v  inet  tcp  /dev/tcp    -
.
:w
"netconfig" [New file] 2 lines, 120 characters
:q
```

Our netconfig can be very minimal, since we need only UDP and TCP. This is one area where the manual page of in.ftpd and empirical data disagree, since the manual page instructs to include ticotsord into the list of networks and copy straddr.so to usr/lib/:

```
ticotsord tpi_cots_ord  v  loopback  -  /dev/ticotsord    straddr.so
```

Note that you would also have to create the corresponding device in the dev/ directory to use ticotsord:

```
# mknod /var/named/dev/ticotsord c 105 1
# chown root:sys /var/named/dev/ticotsord
# chmod 666 /var/named/dev/ticotsord
# ls -l /var/named/dev/ticotsord
crw-rw-rw-  1  root  sys   105,  1  Feb   11 15:05  /var/named/dev/ticotsord
```

Taking these steps and setting up ticotsord does not seem to yield anything, since named works fine without it. Moreover, you should not install a service or a facility into the change-rooted environment if you don't know what it does and especially if it is not really needed by the process you are setting up in the first place. The bottom line is that

there seems to be neither benefit nor harm in setting up ticotsord, which basically renders it unneeded in our change-rooted environment.

Finally, we have to set the proper ownership and permissions for individual files. All files in the directory must be owned by root and set read-only to everyone.

```
# chown root:root *
# chmod 444 *
# ls -la
total 10
dr-xr-x--x  2  root  root  512  Feb 11 15:05  .
dr-xr-x--x  8  root  root  512  Feb 11 15:05  ..
-r--r--r--  1  root  root   75  Feb 11 15:39  group
-r--r--r--  1  root  root  313  Feb 11 15:52  netconfig
-r--r--r--  1  root  root  101  Feb 11 15:10  passwd
```

Note that there is no etc/shadow. It is not required since the change-rooted environment has no login and therefore no passwords to store. Again, this differs from the guidelines for in.ftpd.

5. Set up usr/.
The usr/ hierarchy will accommodate the executables, the required shared libraries and time-zone information. Note that Solaris BIND uses /usr/tmp and BIND 8.2.3 uses /var/tmp. We will create a symbolic link for usr/tmp/ anyway, and the real location for our temp directory will be var/tmp/.

```
# cd ../usr
# mkdir -p lib sbin share/lib
# chown -R root:root *
# chmod -R 551 *
# ln -s ../var/tmp .
# ls -la
total 6
dr-xr-x--x        2 root   root     512  Feb 11 15:55        lib
dr-xr-x--x        2 root   root     512  Feb 11 15:55        sbin
dr-xr-x--x        3 root   root     512  Feb 11 15:55        share
lrwxrwxrwx        2 root   root      10  Feb 11 15:55        tmp -> ../var/tmp
```

The entire change-rooted usr/ hierarchy must be read-only. There are no files or directories in this subtree that would ever change once it has been constructed, and the permissions must be set accordingly.

The best choice to protect the change-rooted usr/ is to have it on a read-only media. This could also be a separate (loopback) file system that has been mounted read-only. One alternative, offered by 4.4BSD,[7] i.e., FreeBSD, is to set the "system immutable" flag for all the files and directories in the change-rooted usr/. An immutable file may not be changed, moved, or deleted. However, the kernel should run in a secure mode to prevent anyone from turning this flag off.

6. Copy the BIND executables to usr/sbin/.
The default installation directory for BIND 8 named and named-xfer is /usr/local/sbin. You only have to change your default directory to usr/sbin/ and copy the executables. These must be owned by root, and the permissions must be execute-only to the world.

```
# cd sbin
# cp -p /usr/local/sbin/named /usr/local/sbin/named-xfer .
# chown root:root *
# chmod 111 *
```

```
# ls -la
total 10
dr-xr-x--x  2 root    root        512    Feb 11 15:05  .
dr-xr-x--x  8 root    root        512    Feb 11 15:05  ..
--x--x--x   1 root    root     619400    Feb 11 14:52  named
--x--x--x   1 root    root     329200    Feb 11 14:52  named-xfer
```

Since we place named-xfer in the change-rooted usr/sbin/, as opposed to the default /usr/local/sbin, we have to specify the new location in named.conf. Otherwise named will not be able to find named-xfer and transfer zones.

7. Set up the shared libraries.
We have to copy the required shared objects—libraries—and ld.so.1, the run-time link editor, into our usr/lib/ in order to be able to execute BIND change-rooted. We will use the ldd command to find out what shared libraries are needed:

```
# ldd ./named
        libnsl.so.1 =>       /usr/lib/libnsl.so.1
        libsocket.so.1 =>    /usr/lib/libsocket.so.1
        libc.so.1 =>         /usr/lib/libc.so.1
        libdl.so.1 =>        /usr/lib/libdl.so.1
        libmp.so.2 =>        /usr/lib/libmp.so.2
        /usr/platform/SUNW,Ultra-5_10/lib/libc_psr.so.1
```

Now we have to copy these libraries and ld.so.1 in our usr/lib/. Since Solaris maintains the specific name (the shared library itself) and a generic name (a symbolic link to the specific shared library) of each shared object, we will use tar to copy these to our environment properly. However, we can simply copy the files that have only the specific name, i.e., the link loader ld.so.1, architecture-specific libc_psr.so.1, and nss_files.so.1.

```
# cd ../lib
# ( cd /usr/lib ; tar cf - libc.so* libdl.so* libmp.so* \
libnsl.so* libsocket.so* ) | tar xvBf -
# cp -p /usr/platform/SUNW,Ultra-5_10/lib/libc_psr.so.1 .
# cp -p /usr/lib/ld.so.1 .
# cp -p /usr/lib/nss_files.so.1 .
# chown root:root *
# chmod 111 ld.so.1
# chmod 555 lib* nss*
# ls -la
total 4094
dr-xr-x--x      2 root    root        512    Feb 6 07:51   .
dr-xr-x--x      4 root    root        512    Feb 6 08:03   ..
---x--x--x      1 root    root     181840    Dec 9 01:05   ld.so.1
lrwxrwxrwx      1 root    root         11    Feb 6 05:51   libc.so ->         ./libc.so.1
-r-xr-xr-x      1 root    root    1015768    Dec 9 01:03   libc.so.1
-r-xr-xr-x      1 root    root      16932    Dec 9 01:03   libc_psr.so.1
lrwxrwxrwx      1 root    root         12    Feb 6 05:51   libdl.so ->        ./libdl.so.1
-r-xr-xr-x      1 root    root       4304    Dec 9 01:05   libdl.so.1
lrwxrwxrwx      1 root    root         12    Feb 6 05:51   libmp.so ->      ./libmp.so.2
-r-xr-xr-x      1 root    root       6732    Jul 15 1997   libmp.so.1
-r-xr-xr-x      1 root    root      19304    Jul 15 1997   libmp.so.2
lrwxrwxrwx      1 root    root         13    Feb 6 05:51   libnsl.so -.       /libnsl.so.1
-r-xr-xr-x      1 root    root     726968    Dec 9 01:03   libnsl.so.1
lrwxrwxrwx      1 root    root         16    Feb 6 05:51   libsocket.so  /libsocket.so.1
-r-xr-xr-x      1 root    root      53656    Jul 16 1997   libsocket.so.1
-r-xr-xr-x      1 root    root      27000    Jul 16 1997   nss_files.so.1
```

The most interesting shared object in our list is nss_files.so.1. It is required by the Name Service Switch, as documented in the manual page of nsswitch.conf.[8] Solaris has three possible databases to store user and password information, i.e., files in the /etc directory, NIS, and NIS+. Since our change-rooted setup uses its local passwd database only, we do not need other nss_* methods from /usr/lib, such as nss_nisplus.so.1. Note that we do not need nss_dns.so.1 either, since the operation of named does not require resolver.[9]

8. Set up the time zones.
The time-zone data files from /usr/share/lib/zoneinfo are needed to report timestamps correctly in logfiles. We only have to copy this hierarchy to our change-rooted usr/share/lib/ and ensure that the ownerships and permissions are properly set.

```
# cd ../share/lib
# ( cd /usr/share/lib ; tar cf - zoneinfo ) | tar xvBf -
# chown -R root:root .
# find . -type f -print | xargs chmod 444
# find . -type d -print | xargs chmod 551
# ls -la
total 8
dr-xr-x--x   3 root     root         512   Feb 6 05:47      .
dr-xr-x--x   3 root     root         512   Feb 6 05:47      ..
dr-xr-x--x  10 root     root        1536   Nov 25 1998      zoneinfo
```

9. Set up var/.
Named must be able to write into the subdirectories of var/. It uses the log, run, and tmp directories for its logfiles, control channel and process id storage, respectively.

Note that var/log/ will not be needed if you decide to set up named to use syslog exclusively. We will discuss the details of this configuration later in the text.

The control channel of named, var/run/ndc.d/ndc is a UNIX domain socket owned by the super-user. It is created before named changes its user-id to bind, and obviously, it is writable by the super-user only. You can use this to your advantage since var/run/ can be set to writable by root, read-only by the group, and unavailable by everyone else.

Finally, we have to keep var/tmp/ for named and named-xfer to write their temporary files. These may potentially all use the available disk space, but at least the risk has been contained in a single directory that can be monitored regularly. You may also want to consider limiting the disk usage of bind by setting up a file system quota.

```
# cd ../../../var
# mkdir -p log run/ndc.d tmp
# chown -R root:bind log
# chown -R root:root run tmp
# chmod 771 log
# chmod 750 run
# chmod 1777 tmp
# ls -la
total 10
dr-xr-x--x      5 root     root       512   Feb 11 06:00      .
dr-xr-x--x      8 root     root       512   Feb 11 07:08      ..
drwxrwx--x      2 root     bind       512   Feb 11 15:06      log
drwxr-x---      3 root     root       512   Feb 11 15:37      run
drwxrwxrwt      2 root     root       512   Feb 11 15:00      tmp
```

10. Configure BIND.

First, we have to change our default directory to /var/named/namedb/ and create the secondary/ subdirectory. This directory will host all the zone files transferred from nameservers that this host is secondary for.

```
# cd ../namedb
# mkdir secondary
# chown root:bind secondary
# chmod 2770 secondary
# ls -ld secondary
drwxrws---  2 root    bind      512   Feb 20 10:26  secondary
```

Set the owner and the group of the subdirectory to root and bind, respectively. Set the permissions to read-write to the owner and the group, set-group-id to the group, and inaccessible to the others. The purpose of the set-group-id mode is to force the group ownership of the files in secondary/ to bind, regardless of the creator of a file.

Now we have our files and directories set, and we can proceed by creating named.conf, the BIND configuration file. In the following example we will only set up a cache-only nameserver that runs in our change-rooted environment. Note that the setup of master and slave nameservers and how to run your DNS are not in the scope of this document: you should consult Albitz & Liu[10] on those topics.

We have to use the options statement to specify the directory and file locations. All absolute pathnames in the following example are in respect to the change-rooted environment. In other words, although the real working directory of BIND is /var/named/namedb, after chroot() the named process sees /var/named as the root directory ('/') and the path to the working directory will be /namedb.

Note that all named-created files will be stored into the change-rooted var/tmp, i.e., /var/named/var/tmp. These files will have bind as the owner and the group.

```
options {
        directory "/namedb";
        named-xfer "/sbin/named-xfer";
        pid-file "/var/tmp/named.pid";
        dump-file "/var/tmp/named_dump.db";
        memstatistics-file "/var/tmp/named.memstats";
        statistics-file "/var/tmp/named.stats";
};
```

We have to set up the named control channel in a safe place. The preferred location, which is subject to file system security on your host, is a UNIX domain socket in the change-rooted var/run/ndc.d. The controls statement sets the owner, and the group of the directory, where the control socket resides (e.g., var/run/ndc.d), to root and read-write permissions to the owner only.

This behavior really is a workaround, since Solaris cannot set permissions for UNIX domain sockets. The permissions are handled properly, for example, on FreeBSD.

```
controls {
        unix "/var/run/ndc.d/ndc" perm 0600 owner 0 group 0;
};
```

We will need only the reverse map for localhost (127.0.0.1) and the list of root nameservers for a cache-only nameserver.

```
zone "0.0.127.in-addr.arpa" in {
      type master;
      file "db.127.0.0";
};
zone "." in {
      type hint;
      file "db.cache";
};
```

If your nameserver acted as a slave to other nameservers, you should store the zone files in the secondary/ subdirectory. As with the files defined in the options statement, the zone files will have bind as the owner and the group. The following fictitious zone is here as an example only:

```
zone "sub.domain.net" in {
      type slave;
      file "secondary/bak.sub.domain.net";
      masters { 169.254.27.16; };
};
```

You can save all log information into the change-rooted var/log/ directory. In our example we maintain three versions and roll the logfile over if its size exceeds 200KB.

```
logging {
      channel local_log {
          file "/var/log/named.log" versions 3 size 200k;
          severity notice;
          print-category yes;
          print-severity yes;
          print-time yes;
      };
      category lame-servers { null; };
      category default { local_log; };
};
```

Storing named log information to the change-rooted area has two problems. From the forensics point of view, it is generally not considered a good idea to leave audit information within the reach of possibly compromised server process. If the change-rooted area is compromised, there is a danger that the attacker could tamper with the log entries.

The management aspect to logging speaks for centralized log control. Since Solaris syslog can collect log information from a change-rooted dev/log device, we only need to decide which log facility to use. In the following example we will use LOG_LOCAL0:

```
# cd /etc
# ex syslog.conf
"syslog.conf" 34 lines, 981 characters
:a
local0.info                     /var/log/named.log
.
:w
"syslog.conf" 36 lines, 1049 characters
:q
# touch /var/log/named.log
# chown root:root /var/log/named.log
# ls -l /var/log/named.log
-rw-r--r--   1 root     root   10453    Feb 23 05:29  /var/log/named.log
```

The last step is to define the channel and the category for logging. Note that we cannot use the built-in default_syslog channel since once a channel has been defined, it cannot be altered. We can still define our own channel and modify the default category to forward all logging to syslog:

```
logging {
      channel local_syslog {
          syslog local0;
          severity notice;
          print-category yes;
          print-severity yes;
          print-time yes;
      };
      category default {
          local_syslog;
      };
};
```

Starting Up

Now that we have our directory hierarchy, files, and devices configured, we can start named change-rooted. This simply means entering the following command:

```
# /usr/sbin/chroot /var/named /sbin/named -u bind -c /namedb/named.conf
# ps -ef | grep bind | grep -v grep
  bind 27621  1  0 17:07:12 ?  0:00 /sbin/named -u bind -c /namedb/named.conf
# tail /var/adm/messages
Feb 11 17:07:12 london.greenwichtech.com named[27620]: starting
        (/namedb/named.conf).
```

You can use ndc to control and stop named, but you have to specify the pathname to the ndc control socket:

```
# ndc -c /var/named/var/run/ndc.d/ndc stop
```

We want named to start automatically during system boot, which means adding the aforementioned command in the RC scripts. Instead of modifying /etc/init.d/inetsvc, the default location to start named in RC, we should keep the original Solaris script intact and write our own RC script that will be executed immediately after inetsvc, aka, /etc/rc2.d/S72inetsvc. Edit /etc/init.d/named as follows:

```
# cd /etc/init.d/
# ex named
"named" [New file]
:a
#!/bin/sh

BINDD="/var/named"
PIDF="$BINDD/var/tmp/named.pid"
NDCHANNEL="$BINDD/var/run/ndc.d/ndc"

case "$1" in
'start')
      if [ -x /usr/sbin/chroot -a \
                -x "$BINDD/sbin/named" -a \
                -f "$BINDD/namedb/named.conf" \
           ]; then
          cd "$BINDD" > /dev/null 2>&1
          if [ $? -ne 0 ]; then
              echo "${0}: ${BINDD}: cannot cd." >&2
```

```
            exit 1
        else
            /usr/sbin/chroot "$BINDD" \
                /sbin/named -u bind -c /namedb/named.conf
            if [ $? -eq 0 ]; then
                echo 'chroot-named starting''
            fi
        fi
    fi
    ;;
'stop')
    if [ -x /usr/local/sbin/ndc -a \
            -r "$NDCHANNEL" -a \
            -w "$NDCHANNEL" \
        ]; then
        /usr/local/sbin/ndc -c "$NDCHANNEL" stop
    fi
    ;;
*)
    echo "Usage: $0 { start | stop }"
    exit 1
    ;;
esac
exit 0
:w
"named" [New file] 39 lines, 715 characters
:q
```

Our RC script checks that the executables and the configuration file are available and then proceeds with starting up named. To stop named, the script checks that the name daemon control program, ndc, and the control channel are available and then requests named to shut itself down.

Please note that although chroot() changes the root directory of your process, it does not change the default directory. To change-root cleanly we should change our working directory to the change-rooted area before calling chroot(). Secondly, the Solaris fchroot() system call can be used to reverse the operation of change-root if there is an open file descriptor to a directory before chroot() was called.

The owner and the group of the named script must be root and sys, respectively. The permissions of the file must be read-write and execute to the owner and read-only to the group and the others. After you have set the ownership and the permissions you must create the symbolic links for the RC directories:

```
# chown root:sys named
# chmod 744 named
# cd ../rcS.d
# ln -s ../init.d/named K42named
# cd ../rc0.d
# ln -s ../init.d/named K42named
# cd ../rc1.d
# ln -s ../init.d/named K42named
# cd ../rc2.d
# ln -s ../init.d/named S72named
```

Now we are all set, and you can start and stop BIND with the RC script. named will start automatically when you reboot the system next time.

Version Control

One common management problem with a change-rooted area is maintaining consistent versions of your binaries and other executables between the change-rooted environment and the rest of the system. For example, when the vendor releases a patch for BIND or the C library, you want to be sure that your change-rooted area(s) will also get updated. On the other hand, you do not want to update the change-rooted executables automatically, without evaluating at least the dependencies first.

You may want to consider your standard UNIX environment as "development" or "staging" and your change-rooted areas as "production." Often there are strict procedures that define how applications and services move from development to production. Accordingly, when you install a vendor patch, it goes to the development/staging area. You may want to set up a secondary change-rooted area for testing purposes and update your production area only after you have ensured that the patched executable or library won't break anything.

To avoid inconsistency in change-rooted production, you probably want to monitor not only integrity of the change-rooted files but also compare them with their counterparts elsewhere in the system. You may not implicitly remember which production files you have to update afterwards, but at least you will be notified.

Conclusions

The text has described how to set up and run BIND 8 change-rooted on Solaris and more specifically, on Solaris-2.6. This is the most protective setup available on UNIX, and the same methodology can be applied to change-rooting other UNIX services as well. The author has been running change-rooted BIND 8.2.3 on Solaris and FreeBSD since the release of the recent CERT advisory.[11]

Generally speaking, it is unwise to run any super-user-owned network service on your machine, especially if you haven't taken any steps to protect the operating system from an attack. From the practical point of view, there is no reason why you should not change-root your named, since, requiring few external resources, it is almost an ideal program to be confined.

The most significant threat to running named or any other UNIX process change-rooted is the possibility that the attacker might be able to revert back to the system root. For this reason a change-rooted process is never completely safe if it is owned by the super-user.

REFERENCES

1. S. Burr, How to break out of a chroot() jail, *http://www.bpfh.net/simes/computing/chroot-break.html*, January 31, 2001.

2. W. Venema, Chrootuid-1.2, *ftp://coast.cs.purdue.edu/pub/tools/unix/sysutils/chrootuid/*, August 16, 1993.

3. H. Pomeranz et al., *Solaris Security: Step-by-Step*, Version 1.0, The SANS Institute, 1999.

4. Internet Software Consortium, BIND 8.2.3 source package, *ftp://ftp.isc.org/isc/bind/src/8.2.3/bind-src.tar.gz*, January 26, 2001.

5. Sunfreeware.com, BIND 8.2.3 Package for SPARC/Solaris8, *ftp://ftp.sunfreeware.com/pub/freeware/sparc/8/bind-8.2.3-sol8-sparc-local.gz*, January 29, 2001.

6. Sun Microsystems, Inc., *in.ftpd – Internet File Transfer Protocol server*, Solaris System Administrator's Reference Guide, March 4, 1997.

7. M. K. McKusick, et al., *The Design and Implementation of the 4.4BSD Operating System*, Addison-Wesley, 1996.

8. Sun Microsystems, Inc., *nsswitch.conf – configuration file for the name service switch*, Solaris System Administrator's Reference Guide, April 28, 1997.

9. Sun Microsystems, Inc., *resolver – resolver routines*, Solaris Programmer's Reference Guide, December 30, 1996.

10. P. Albitz, C. Liu, *DNS and BIND*, 3rd Edition, O'Reilly & Associates, September 1998.

11. CERT® Coordination Center, CA-2001-02: Multiple Vulnerabilities in BIND, *http://www.cert.org/advisories/*, January 29, 2001.

get my goat

by Steve Johnson

Steve Johnson has been a technical manager on and off for nearly two decades. At AT&T, he's best known for writing Yacc, Lint, and the Portable Compiler.

yaccman@earthlink.net

and Dusty White

Dusty White works as a management consultant in Silicon Valley, where she acts as a trainer, coach, and troubleshooter for technical companies.

dustywhite@earthlink.net

Organizations, in many ways, act like living creatures. The shared feelings and beliefs in a company, or even in a department of a company, can sometimes take on a life of their own. Biologists describe something as alive if it has recognizable boundaries and takes active steps to preserve the integrity of those boundaries. In this and the next couple of articles, we will discuss how this works with organizations.

If a living thing picks up something it can't ingest (say, a splinter), the first response is typically to isolate it first (for example, grow a protective layer of tissue around the splinter) and then expel it. In the case where we eat food that is tainted, the first step is skipped – we just get rid of what we can't digest, sometimes violently.

Organizations are pressured to digest people, ideas, and technology. Sometimes they cannot. So organizations act very much like other living things: they isolate, and then expel, what they cannot integrate. At its best, this is a very natural response that keeps the organization healthy. At its worst, it is a prelude to death throes.

We believe that these patterns of behavior are very deeply ingrained in us. They are not necessarily irrational, but they are certainly a-rational. They take place below the level of rational thought. As a result, symbolism, so beloved by the unconscious mind, is often employed by an organization's "immune system."

One very clear way to see this is the phenomenon of scapegoats. The term comes from the Old Testament Jews, who, once a year, would have a ceremony where the sins of the entire tribe were symbolically transferred to a goat, who was then driven out into the desert to die. So this behavior is wired into our traditions for several millennia.

Nearly everyone who has worked at a company for a few years has seen scapegoats. A project fails, money or business is lost, there is a period of confusion. Then somebody (typically two managerial levels below the real source of the problem) is identified as the cause of the debacle. Depending on the company culture, the person may be publicly humiliated, be forced to resign, be fired, or spend several years "in the penalty box" where they are given postings to Siberia, or put in charge of preparing the budget. Managers are often defrocked or their organization is gutted. Technical people are more likely to be let go, or promoted to high-sounding but meaningless titles ("Corporate Technology Guru") and ignored.

When this happens, the majority of partly guilty people can relax – their sins have been forgiven, handed to the scapegoat, and they can get on with running the organization. This is actually healthier for the organization than recriminatory wallowing in its failures, so organizations that have a well-developed immune response tend to succeed over the long run. Large, successful companies often have carried scapegoating to a high art form. Sometimes it is the scapegoat's personality that causes the problems. Sometimes it is poor business or technical judgment. Often, the most virulent and irrational scapegoating involves ideas.

Say a new employee, call him Smarty, arrives and quickly starts saying loudly to everyone in sight that the current project should be scrapped and rewritten in Java. The immediate effect of these pronouncements is to make the current staff, all C++ experts, feel stupid. Nobody likes to feel stupid. Moreover, Smarty is criticizing the plans of the organization, increasing the stress level of both managers and programmers. In effect, the organization begins to run a fever. Sometimes, the new idea infects a few people and spreads through the organization. (Depending on the idea, this may be a good or bad thing!)

But sometimes the idea is too uncomfortable. The organization will isolate it and then reject it. A key manager makes a couple of suggestive comments, and those in the know realize that Smarty's days are numbered. People stop wanting to sit next to him at lunch. Key pieces of information just don't seem to make it to Smarty's desk. He gets more and more uncomfortable and may not even realize why. Eventually, if he doesn't get so uncomfortable he leaves, he will be transferred or fired. Then the organization will heave a collective sigh of relief (ignorance being bliss, after all). The stress level goes down, and the organization actually functions better (albeit they may still be doing the wrong thing).

If Smarty is a manager, the projects he championed may be cancelled. In one case we know of, the manager left, and the project he had protected against great opposition was almost immediately canned. Three months later, it was resurrected under another name. A year after that, it was the biggest income producer for the department. When the original manager pushed the project, the organization couldn't "digest" it. After the sins were laid on this manager and he left, the "infection" left with him. That allowed the organization to see the positive things about the project, and find a way to digest it, under a different name.

You say, is this fair? Of course not. Bearing the sins of an entire organization is hard work. Scapegoats are rarely happy people and may even develop serious health problems from carrying all those sins that aren't theirs. Perhaps they can gain some solace by thinking that some people who died to save others from their sins are pretty fondly remembered by various religions. Fair or not, it does seem to be inevitable, wired into our culture if not our neurology.

Bearing the sins of an entire organization is hard work.

Clearly, it is best not to become a scapegoat. If you encounter resistance to your ideas or your style, be aware of your effect on other people. Watch particularly for any sign that people are feeling put down or inferior to you, because this generates strong negative energy in most people.

Sometimes, however, you must push for what you believe is right, even if it gets you in organizational hot water. In one case we are aware of, an individual took a stand that he knew would get him scapegoated because he honestly believed that this was the best thing for the company (in which he held many stock options).

He indeed was scapegoated; the organization came together and, in fact, took the path he had espoused, and the company went on to become successful. He was, in effect, paid several million dollars to be a scapegoat (at least, that's how he sees it now). Considering how painful the experience was, he doesn't feel overpaid.

In a large company, a physical transfer to a new location is often the best way to recover from scapegoating. (One scapegoat I know took a posting to Japan. When someone asked his boss why he was sent to Japan, a co-worker quipped, "We don't have an office on the moon yet.") He returned in triumph several years later – the beliefs that had got him exiled had finally won the day in his old organization, and his recent successes in Japan had given people more respect for him. He was out of the penalty box.

For many people, however, the best thing to do is to learn what you can from the experience of being scapegoated, and then leave. Pick your next company wisely. Watch the effect you have on others. Make sure you have some allies – peers and managers – before trying to make big changes in the organization. And when scapegoats are created (and they will be), show some compassion. It could easily have been you.

contractors or employees? it's not that simple!

by Strata Rose Chalup

Founder, VirtualNet Consulting; Strata Rose Chalup specializes in large-scale messaging deployment. A sysadmin since 1983, she is now enjoying a sabbatical to scuba dive, read sci-fi, fix her house network, and get enough sleep.

strata@virtual.net

The New Moving Target

The recent shakeout in parts of the high-tech sector has had some interesting effects on employment patterns, especially here in the Silicon Valley area. I've watched long-term employees become contractors, and folks who have been contracting decide to "hunker down" and become employees. I've also been watching the inflammation of the usual mythology about employees vs. contractors on projects and think it's time to share some observations in the hope of increasing the light-to-heat ratio.

I'll say up front that I've been primarily a contractor for many years, and am a "career" contractor with no desire to become an employee at a typical company. Project managers reading this will not be surprised to hear that I have found good project management to make more of a difference than the employee-contractor ratio on a project. This has been the case with every project in which I have ever participated, either as an individual contributor or as the project manager, in an employee role or as a contractor. For those who have been burned by excessive rote-oriented project management, I hasten to add that the quality of project management is often *inversely* proportional to the amount of paper (virtual or otherwise) generated by the approach. One consequence of the recent market roller coaster is that many companies are short-staffed, have scarce financial resources, or both. Yet, as always, the amount of work needing to be done hasn't decreased! New projects with short turnaround times are popping out of the woodwork at many firms as companies struggle to adjust to changing conditions and requirements. You probably just want to get things done, but you have to figure out the best way to do it. Traditional solutions tend to involve bringing in teams of contractors or scraping up a few die-hards on staff to overload. There are a lot more options out there, but to understand how to use them, it's useful to take a quick survey of some of the major types of contributors. Armed with this knowledge, we may be able to put together teams that will leverage dwindling financial and attention-span resources in a more effective way than usual.

It is never good practice to rule out someone, or to guarantee their presence on a team, based solely on their work orientation.

Types and Tendencies: A Guideline, Not a Rule

For the past several years in particular I have been involved in team-oriented project work, to carry out specific large e-commerce or IT build-outs (30K IT user to 500K ISP user rollouts) from 3 to 14 months in duration. I've been a team member, team leader, have constructed teams, and have been handed already-formed teams. My observations are primarily in the area of project-oriented work, rather than long-term "virtual employee" work. I believe that the insights have validity in both situations. In my experience, I have found that significant, repeatable differences exist between various types of contributors. Obviously, individual dedication is a major factor – you will always meet people who are the exception to the general rule, either in a positive or negative direction. The last thing that I want to do is create a new set of knee-jerk mythologies! It is never good practice to rule out someone, or to guarantee their presence on a team, based solely on their work orientation. The primary goal of this article, in fact, is to encourage people to reconsider the idea that either contractors OR employees are always better.

Individuals reading this article may recognize themselves in one of the types described, and should take two specific messages away from this article. The first is that people generally move through various states in their careers, whether they are employees or contractors. These states are not necessarily progressions from one stage to another, but are more about the person's focus at that point in their career. The second message is that each state tends to have specific strengths and weaknesses associated with it. Persons concerned with improving their job skills can use these descriptions to celebrate and improve their strengths, and work on correcting their perceived weaknesses. Those who feel that their particular work style is in itself an exception to the rule will, I hope, come away with a better understanding of the prejudices that may work against their participation in various projects. They can also look at some of the positive aspects of other styles of contribution, and choose to develop some of those capabilities, regardless of their focus.

The sloppiest work I routinely encountered was done by employees who were just biding their time to early retirement, in what is generally referred to as "rest and vest."

Here, But Not Here

The sloppiest work I routinely encountered was done by employees who were just biding their time to early retirement, in what is generally referred to as "rest and vest." Many were in their final year of a four-year vesting, or were coming up to a significant vesting shelf. These folks are often just concentrating on not getting fired, and it's like pulling teeth to do anything that will affect them. Now that the market has tanked, we may see less of this kind of attitude, and that's just fine with me. Of course, not everyone who is waiting to cash out has the "throw-away" attitude. Until recently, however, there were enough of these cases that you were almost certain to run into them in various industry sectors. Let me hasten to say that I don't want to tar with the same brush those folks who have decided that their current job and their future career are a bad match, and who have made up their mind to leave. These folks are usually trying very conscientiously to leave with a clean slate and a good impression. Ditto for the majority of older employees awaiting retirement, early or not, who generally have a lifetime of work habits that keep them following through.

New Kids on the Block

Next in quality were the junior contractors who hadn't learned the ropes yet, independent or not, and the junior or intermediate ones who came from a generic technical consulting shop. You know the type – on hourly at a good rate as W2 subcontractors via an agency, and generally "hire and forget" in terms of direct supervision once they pass the resume and phone screen. If you have high standards as a project manager, you will find that many (not all, but many) of these folks will take extra effort to handle. This subset is not used to being held to certain levels of documentation and accountability and may need substantial education on change control. If they are new to contracting, or new to the industry itself, their business skills will almost certainly need some attention, even on simple issues like communication and follow-through. The fact that they are usually dedicated to your project is a substantial asset that by itself can overcome some of the liabilities.

Many projects will bring in "generic" contractors as extra manpower to achieve specific ends, rather than as problem solvers. Junior contractors may not realize this. Many of them entered contracting as a way to keep troubleshooting or design skills prioritized in their careers. A distressing number of them have a bit of a chip on their shoulders, thinking of themselves "the smart guys/gals who were brought in because nobody here could do the job." As such, they often feel they have carte blanche to make or suggest changes that can be disruptive of project success and overall client relationships. The

specific role of junior contractors should be made clear to both the agency and the contractor during the screening process.

Systems Ronin for Hire

These are new kids on the block after they've been around the block a few times. They come in, get the job done, and go home again. Occasionally, they are marking time while waiting for a better assignment to come along and have to be nudged a bit to make sure that t's are crossed and i's are dotted. Sometimes they are independent, but most often they are from contract agencies. They are not yet oriented as "career contractors" but are very definitely focused on contracting. They have usually picked up a good set of business skills and have a solid enough technical background that the agency always has a new gig for them. A significant minority have an entrepreneurial bent and may be in contracting while keeping an eye on start-up opportunities. A rare few may need to be explicitly reminded that their job is to do what they were hired for, under your direction, not to invent new work for their agency or go off on tangents. These can be both the best and the worst of the crop: the worst are the people who don't like what they are here to do and are looking for something more interesting; the best are the career-minded folks who are just starting out and are genuinely trying to build the business for themselves and for their agencies. If they actually discover serious infrastructure flaws at the site, encourage them to have their agency submit a proposal to deal with the situation. In any case, they should not consider themselves free to "fix" the site infrastructure beyond the scope of their job or project duties, even if the time spent would not negatively affect the project.

Joe or Jane Employee

Here we have our standard-bearers, the folks who make the wheels turn and the sun rise – regular employees who are *not* planning an imminent change in career or a cash-out on their options. They understand the need for quality, and are generally pro-active about things like documentation, change control, and good planning. They understand that they will be taking the reins once the project is in sustaining mode and the contractors leave. In addition, they usually bring in valuable experience of the organization itself – how to obtain resources, who to contact to resolve difficulties, and so on. They are generally integrated into the existing structure of the organization or business, and can apply leverage exactly where it's needed to get the job done and move things along.

This integration can be a double-edged sword. Employees on a team are likely to directly report elsewhere and have substantial other responsibilities, often to basic infrastructure commitments which supersede an individual project. The selective availability of employees as a project or team resource needs to be realistically balanced against the demands of the project. It is vital to come to an understanding with employees' management about their role in a project, and achieve consensus on the level of time and effort that will be committed by the employee. It is also an excellent idea to make sure that the employee does not feel "drafted" to participate in the exchange and that management's idea of the employee's time availability for a project is actually grounded in reality.

Wildcards/Moonlighters

Each individual of this type is a case unto him or her self. That is the "moonlighter" contractor, the guy or gal who is doing a contract for a few months while between jobs, or to pick up a little extra income, or to see if they want to get into contracting full-time. This is not uncommon in programming or IT work. These folks range all over the map in technical seniority – 2 years to 5 years to 10 years! If they are older, they are more

likely to be making a move into full-time contracting. If they are younger, well, that's less likely. They are often receptive to offers of an employee position, and may be using contracting as a way to "test-drive" your company before settling down. Perhaps they've had an unpleasant experience elsewhere.

So – what about moonlighters? These folks can be a godsend, turning into employees who will provide continuity and settle down, or they can be the "fast and loose" type who come in, razzle-dazzle the problem, but leave an undocumented and unmaintainable solution. In the worst case, they may get a job offer somewhere else and leave before the contract is up! When you evaluate contractors' resumes, make sure to get the skinny on which positions were contract and which were employee. Evaluate the person on their merits, consider their work history, and figure out if their involvement works for you.

Systems-Team Contractors

Here you often find the employees of project-oriented contracting firms, who are on salary rather than on hourly pass-through. They are contracting on behalf of their parent organization and make up part of a virtual team of sysadmins working together. These folks usually combine the best qualities of regular employees and career contractors. They are conscious of the need to maintain the brand image of their employer, and they work with a higher degree of professionalism than most agency contractors. One advantage of using a "systems team" contracting firm is that, unlike "body shop" contract firms, the team-oriented sysadmins are often explicitly encouraged to call upon each other as backup on contracts. The best firms employ a substantial amount of group infrastructure to encourage this behavior, including mailing lists for hot issues, databases of past problem solutions, and the like.

A successful systems-team shop probably has several employees of nationwide professional stature whose expertise in total spans several important categories in system administration, such as backups, storage area networks, or high-performance servers. These employees serve as architectural and design resources for the whole group, and can represent a great leveraged resource for your project.

Career Contractors

Career contractors have a long-term reputation to protect and are accustomed to being, if not more careful, then perhaps more thorough than the regular employees. Documentation gets finished at the end of the project, changes are checked against the design or the management, specifications get written before implementation starts. A good contractor realizes that he or she has all the responsibility, but none of the decision-making ability, of the organization employee, unless such ability has been explicitly delegated. The best contractors are aware of how much the worst ones poison the well for all contractors, and will work extra hard to combat that image and build confidence in the contractor community. Career contractors, incorporated or not, are committed to what some call "the brand of Me" and see each project as a chance to build that brand image by delivering high-quality work. Most of their jobs come from word-of-mouth referrals and repeat business, so they have strong motivation to do well.

Independent career contractors are likely to be senior in technical skills, with 10 to 15+ years of industry experience, and also to possess commensurate business skills from having to interface with many companies and teams over their career. They generally have very focused work habits, and are producing real work for you throughout most of the eight hours billed, unlike most employees. They aren't cheap, but they do provide

amazing value. The primary caveat with career contractors, particularly independents, is that they may have substantial responsibilities to other clients. It is highly desirable to get a specific commitment of time and resources for the projected period of the project.

Rising Stars

At the very top of the quality scale we have the career-track employee who is planning his or her rise within the company. This person is usually a superstar. A super-duper star most times! This is the person whose reputation is responsible for the myth that employees are always better than contractors. He or she is motivated, has the same quality agenda as a senior independent contractor (the brand of Me), and is willing to burn the candle at both ends and in the middle to deliver as platinum-plated a result as possible. If you find one of these folks in your organization, you had better have an escalating series of responsibilities to hand out. He or she is focused on the climb to the top, either at your company or somewhere else. You have a motivated resource without the usual time or budget constraints, since this person will give up their personal time to succeed and they are on salary. Do the rest of us a favor: please don't take too much advantage of this and burn the poor guy or gal out!

Match the Person to the Project

So – what's the best ratio of employees to contractors on a project? It really depends on how you classify the folks you have to work with, and what you can get. Trying to identify a standard figure like 20/80 or 30/70 or 40/60 can produce a great project or a disastrous project, depending on who participates. I'd say that the key would be to figure out what resources you have within the company and carefully build the ones you don't have, being careful to match types and individuals with the roles you have to carry out the project. For instance, if you have a superstar employee who can provide careful mentoring and management, or you are managing just the one project and can do it yourself, you could fill in more with "Joe BodyShop" type contractors. On the other hand, suppose you have mostly junior employees, or a substantial percentage who are getting itchy feet for whatever reason. In that situation, you'd be better off making sure you have at least one highly senior contractor, because he or she will help keep the project on track and actually provide more continuity than employees who are on their way out the door. For many projects, a bit of digging in your own organization may preclude the need to hire contractors at all – approach management with a strong plan, and you can often pick up a portion of people's time, especially if your department or group is willing to kick in what would have been contractor money toward overhead on the "loaner" project members from other departments.

More than One Right Answer

As you can tell by now, there is definitely more than one right answer to the question of employees vs. contractors. Successfully staffing any project requires an understanding of "people issues" as well as of the technical specifications of the project. By paying attention to both of these factors, it is possible to create a team mix more closely tailored to the project requirements. Examining some of the work styles and motivations of both employees and contractors in different stages of their careers shows us that there is considerably more flexibility possible in staffing most projects than we might have previously believed.

the bookworm

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He is Editorial Director at Matrix.net. He owns neither a dog nor a cat.

<peter@matrix.net>

Everyone knows that I get and read a lot of books. Frequently, I look at books I dislike. By and large, I try to avoid really negative reviews. This column is devoted to a negative review because I fear that this book will be well-regarded because it originates from a major publisher.

Who's on the phone?

I'm at a complete loss as to why McKnight, Lehr, & Clark's *Internet Telephony* was published. It is rife with outdated "facts," misleading in import, poorly proofread, and (in general) an embarrassment to its publisher.

The volume comprises 13 chapters, many of which are by the editors' students. "An Introduction to Internet Telephony" (1-13), is nothing of the kind. It flatly notes that "If a reader is hoping to unravel the mysteries of, for example H.323, H.324 . . . , we suggest that person look elsewhere"[p. 5]. (This appears to be the sole mention of the ITU's standards.) The editors further note that "We define Internet Telephony as the services, applications, and equipment for mediated human communication emerging from the convergence of the Internet and telecommunications. That is the subject of this book" [p. 7].

I found little of that subject elucidated.

I was looking forward to Dave Clark's "A Taxonomy of Internet Telephone Applications" [pp. 17-42]. Here, I was rewarded. Clark is the author (or co-author) of about 15 RFCs, most interestingly of RFC 1633 (June 1994) on "Integrated Services." In "Taxonomy," he uses some of this information, as well as material in RFC 2205 (September 1997) and 2211 (September 1997). It's unclear to me why RFC 2750 (January 2000), which updates 2205, isn't cited, unless this volume has been in production for over a year. One of Clark's great assets is his view that the "various industry players" will find it increasingly incapable of agreement, necessitating resolution "in a multinational context" [p. 41].

McKnight's and McGarty's essay on "Virtually Global Telcos" [pp. 43-91] informed me that "the World Trade Organization (WTO) has superseded the ITU as the organization setting the terms for international telecommunications services" [p. 68]. While it is true that the CCITT was subsumed into the ITU, this led to the formation of ITU-T, the "Telecommunication Standardization Sector." Those of us who follow H.320, H.323, etc., would be surprised by this cis-Atlantic chauvinism. I found it yet more fascinating as McKnight and McGarty repeatedly cite SS7 (Signaling System 7), which is a 1980 CCITT standard which was detailed in an AT&T paper at the Spring 1987 International Switching Symposium. The business concepts within the essay may be valuable, the misleading statements vitiate them.

Lehr's piece on "Vertical Integration" [pp. 93-124] would have made a good 10-pager. But it is rife with platitudes like: "There are strong incentives to integrate vertically (and horizontally) for each of the participants in the service provider value chain" [p. 112]; and: "Technological advances have been reducing network costs in absolute terms. . . ." [p. 123].

I was looking forward to Dave Clark's "A Taxonomy of Internet Telephone Applications" [17-42]. Here, I was rewarded.

BOOK REVIEWED IN THIS COLUMN

INTERNET TELEPHONY

L.W. McKnight, W. Lehr, and D.D. Clark, Eds

Cambridge, MA: MIT Press, 2001. Pp. 395.

ISBN 0-262-13385-7.

Clark is the author (or co-author) of about 15 RFCs, most interestingly of RFC 1633 (June 1994) on "Integrated Services." In "Taxonomy," he uses some of this information, as well as material in RFC 2205 (September 1997) and 2211 (September 1997). It's unclear to me why RFC 2750 (January 2000), which updates 2205, isn't cited, unless this volume has been in production for over a year. One of Clark's great assets is his view that the "various industry players" will find it increasingly incapable of agreement, necessitating resolution "in a multinational context" [p. 41].

McKnight's and McGarty's essay on "Virtually Global Telcos" [43-91] informed me that "the World Trade Organization (WTO) has superseded the ITU as the organization setting the terms for international telecommunications services" [p. 68]. While it is true that the CCITT was subsumed into the ITU, this led to the formation of ITU-T, the "Telecommunication Standardization Sector." Those of us who follow H.320, H.323, etc., would be surprised by this cis-Atlantic chauvinism. I found it yet more fascinating as McKnight and McGarty repeatedly cite SS7 (Signaling System 7), which is a 1980 CCITT standard which was detailed in an AT&T paper at the Spring 1987 International Switching Symposium. The business concepts within the essay may be valuable, the misleading statements vitiate them.

Lehr's piece on "Vertical Integration" [93-124] would have made a good 10-pager. But it is rife with platitudes like: "There are strong incentives to integrate vertically (and horizontally) for each of the participants in the service provider value chain" [p. 112]; and: "Technological advances have been reducing network costs in absolute terms . . ." [p. 123].

Clark, Again

Dave Clark's "Local-Loop Technology and Internet Structure" [pp. 125-140] is a diamond among the coal. Though it's a version of a paper that appeared two years ago, Clark has revised and emended it. Clark really understands the problems facing the LEC as well as the capabilities of the LECs' wire lines. He concludes that "there may be increased competition in the provision of [consumer] services . . . This derives from the open character of the Internet design that militates against vertical integration of the Internet service provider and the higher-level service provider" [p. 139]. Dr. Clark, meet Dr. Lehr.

That's the first five or 13. I'm not going to savage each of the others (sorry). The one by Mutooni and Tennenhouse [pp. 143-163], for example, is an excellent presentation on "Internet Telephony and the Datacentric Network." But McKnight and Shuster, "After the Web" [pp. 165-190], is full of weird statistics and graphs.

Let me start with a look at the growth of the Internet. On p. 175 there is a graph of Internet hosts attributed to Network Wizards showing 0 in January 1969 and 30 million hosts in January 1997. The 30 million was reported by NW in January 1998. The graph on the next page (no attribution) levels out predicting under 100 million hosts from January 2002 through January 2011. Using the information from Network Wizards and NSI, we passed 100 million hosts last November.

The bibliographical references are either misspelt ("Lotter" for Mark Lottor) or absent. Talking about numbers of Internet hosts or Internet host growth without a reference to John Quarterman is absurd. Referring to a 1997 (dead) URL (Hilgemeier) for "Internet Growth" is not useful.

Etc.

I'm very unhappy about this book. My suggestion is that if you're interested in Internet Telephony, get Hersent, et al. *IP Telephony* [Addison-Wesley, 2000]. The few good articles in this volume aren't worth $40.

book reviews

LINUX ADMINISTRATION: A BEGINNER'S GUIDE

Steve Shah

New York: McGraw-Hill, 2001. Pp. 643 + CD-ROM. ISBN 0072131365.

Reviewed by Paul Guglielmino
<*paulg@ccs.neu.edu*>

[Paul Guglielmino is a UNIX administrator in Boston.]

Steve Shah's *Linux Administration: A Beginner's Guide* is a well written and comprehensive book on Linux administration. The book is divided into seven-sections: installing Linux as a server, single-host administration, Internet services, intranet services, advanced Linux networking, and two appendixes. Some specific areas dealt with are software installation, the bootup and shutdown process, Samba, DHCP, and backups. Internet services covered are SMTP with Sendmail, DNS, FTP, and Web services with Apache. This book is written for beginners, but there is a section about some of the more advanced features of the Linux kernel. This includes, among other things, IP masquerading, IP chains, packet filtering and the /proc file system. Unfortunately, in trying to cover as much ground as he has, Shah can't delve into the details of any one subject, but I think he does a good job of giving a clear overview of each topic without overwhelming new users with details.

Even though the book does not have "RedHat" in the title, it is obviously centered on the RedHat distribution. There are several references to RedHat in the text; more specifically, the installation chapter only describes the installation process for RedHat. In fact, other distributions are only mentioned in a few sentences in the opening chapter. The book comes with a CD, which has a watered down RedHat 7 distribution.

Although very good overall, the book could stand some improvement in a few places. My biggest complaint is that the security chapter could have been developed more. There was no mention of TCP wrappers or the importance of good passwords. (On the plus side, there was a whole chapter dedicated to SSH.) The installation chapter could have said more about dual booting Linux and Windows, since this is an issue that could potentially scare away new users of Linux. I was also a little disappointed in the chapter devoted to the Linux kernel. Since this book is about Linux administration, I would expect this chapter to be one of the most thorough. Although Shah does explain many of the different options that are available in the kernel and then goes on to show you how to compile your own, he could have described the history of the kernel and the kernel module system in more detail.

In addition, the primary discussion of LILO occurs in an earlier chapter, but it would have been more useful to cover LILO in the context of the Linux kernel. Since the book is RedHat-centric, showing how to install the kernel via RPMs would have been helpful. Another small complaint is that there is a small chapter on POP but no mention of IMAP anywhere.

The introduction states that the reader should be a "strong user in Windows." The best part of the book may come in the first chapter in a section titled "The major differences between Windows 2000 and Linux." Here there are discussions of the separation of the GUI from the kernel, single and multi-user philosophies, the Windows registry versus text-based configuration files, and Active Directory vs. NIS. Other than that great section and a couple pages of blueprints about the boot and shutdown processes, there is not much reason to be well versed in Windows to read this book. I liked that the software installation chapter explained both how to use RPM and how to compile packages from the source. RPM is a great tool, but it's important for a system administrator to understand what her tools do and how they do it. The SMTP chapter was a very good introduction to Sendmail, which can be a very complex program to administer. The chapter was Sendmail-centric, but that doesn't matter much because it is by far the most common MTA on the Internet. In each of the "Internet" chapters, Shah has included a mechanics section that could be considered a very condensed version of the relevant RFC. It is important for troubleshooting to know not only what software does but also how it does it. Each chapter has a nice set of summary bullets, and most chapters include a small list of useful links or book references to learn more.

I would definitely recommend this book to those new to the world of Linux and UNIX. However, if you have some experience in UNIX, but are new to Linux, I would advise checking into other Linux books on the market or just looking to the Linux Documentation Project on the Web for help.

DATA MUNGING WITH PERL

David Cross

Greenwich, CT: Manning Publications Co., 2001. Pp. 283.ISBN 1-930110-00-6.

Reviewed by William Annis
<*annis@biostat.wisc.edu*>

With entire rows of bookstores full of books on learning CGI programming with Perl in 10 easy lessons, it's nice to see one highlighting Perl's data processing capabilities more generally.

The book is divided into three parts with several chapters apiece. The first part is an overview. After defining "munging" and introducing his CD collection file – which will show up in examples throughout the rest of the book – Cross discusses Perl itself then goes on in Chapter 2 to discuss important issues like separating parsing from munging, choosing the right data structure for your problem, and approaching audit trails and data validation.

Chapter 3, "Useful Perl Idioms," gives an excellent overview of sorting in Perl, including a description of the Orcish Maneuver and a very nice explanation of not only how the Schwartzian Transform works but also why you'd use it. Tools for debugging and benchmarking are introduced in this chapter, as is the DBI interface. Chapter 4 goes over important string manipulation functions and regular expressions.

Part 2, "Data Munging," is the heart of the book. Starting with simple transformations of ASCII data to other formats, Chapter 5 ends with a discussion of several CPAN libraries for manipulating and formatting numeric data. Chapter 6, "Record-oriented Data," starts with a three-page digression on the while (<FILE>) {} idiom, which seems a bit strange coming so late in the book. It goes on to various types of line-oriented records and introduces useful idioms using the Perl special variables which apply to records ($", $/, $, etc.). Data caching, CSV, and multi-line records are touched on, and Chapter 6 ends with a good overview of date and time manipulation, including parsing dates, using POSIX::strftime, Date::Calc, and Date::Manip. Chapter 7, with a discussion of various sorts of fixed-width and binary data formats, ends Part 2.

Chapter 8 starts off Part 3, "Simple Data Parsing," by introducing more complex data files and metadata. There is a brief section showing how regular expressions are not sufficient to parse HTML, followed by an introduction to parsing terminology. Chapter 9, "HTML," builds a few tools for summarizing and manipulating HTML using the CPAN modules HTML::Parser, HTML::LinkExtor, and HTML::TokeParser. The chapter ends with an example for getting a weather report from Yahoo!. Chapter 10, "XML," starts off with a quick introduction to XML and makes a clear distinction between valid and well-formed XML. The chapter is only concerned really with well-formed XML, so there is no discussion of DTDs. After discussing several modules for XML parsing – XML::Parser, XML::DOM, and XML::RSS – Cross ends with a tool for turning XML markup of documentation into POD, HTML, and plain text. Chapter 11, "Building Your Own Parsers," gives a nice description of Parse::RecDescent. After introducing parser-writing using Windows INI files as an example, the chapter ends with a parser for Cross's CD collection file.

The emphasis on using existing tools from CPAN is a strong feature of this book. Appendix A gives brief documentation on the modules used in the main text. Finally, Appendix B is a very dense and brief overview of Perl itself. This, with the Perl boosterism early in the book, is a bit odd. No one unfamiliar with Perl is going to be able to use this book as a recipe book, but for experienced Perl programmers it is an excellent overview of Perl's many data manipulation capabilities.

DNS AND BIND, 4TH EDITION

Paul Albitz and Cricket Liu

O'Reilly & Associates, Sebastopol, CA, 2001. Pp. 622. ISBN 0-596-00158-4

Reviewed by Rik Farrow
rik@spirit.com

I have the second edition of this excellent book, but remembered a couple of small problems I had when I first examined *DNS and BIND* back in 1997. For example, I only dimly understood the meaning of "canonicalization," and I really wanted Albitz and Liu to explain the term in their book so I wouldn't have to ask someone and appear clueless. The second edition just assumes that you know what the canonical name means, but the fourth edition actually defines it on page 8. Thank you, guys.

Of course, that's not the biggest change in a book that has grown by 200 pages in two editions. A lot of the book has been revised "for the first time," and that has made the new edition easier to understand. The explanations are a bit longer and are certainly clearer than before.

The other big difference is that the fourth edition includes both version 8 and version 9 features, something my old edition misses entirely. So, you can learn about how to configure BIND to handle IPv6 addresses, and to use transaction signatures. If you are using version 9, you can even use DNSSec, a method that will quadruple the size of your zone files while adding security that only other participating BIND 9 servers will appreciate. Mind you, I do think that DNS does need better security, and anxiously await the outcome of the ongoing experiments with DNSSec in The Netherlands, Germany, and Sweden. Evi Nemeth has a new chapter in her sysadmin book about DNSSec, as well as a version of that information in the last special edition of *;login:*. Even so, it helped having yet another explanation of a complicated topic.

Overall, I liked the changes and improvements to the fourth edition and can recommend it to anyone who uses BIND. I think the authors have done a great job in improving this book and deserve to be congratulated.

standards reports

Our standards report editor, David Blackwood, welcomes dialogue between this column and you, the readers. Please send your comments to <*dave@usenix.org*>

Update on POSIX® Test Methods

by Barry Hedquist

Barry Hedquist is the chair of the IEEE POSIX Test Methods Working Group and works for Perennial,

beh@peren.com

As the POSIX era, as we know it, transitions to a new paradigm, it may be worth updating the state of standards that define testing for POSIX conformance.

General Test Methods

IEEE Standard 2003:1997 and the ISO version, ISO/IEC 13210:1999, cover the general requirements for testing conformance to POSIX standards. This document describes how to write the assertions used to specify POSIX test methods, the minimal requirements for a POSIX Conformance Test Suite, and the general criteria for assessing conformance of an implementation to any of the POSIX standards. It provides requirements and guidance to those who write POSIX test methods, develop POSIX test suites, or conduct POSIX conformance testing.

POSIX.1 Test Methods

The test methods for the 1990 version of POSIX.1 (System Interfaces) are contained in IEEE Standard 2003.1-1992. This standard was recently reaffirmed by IEEE, and even more recently published by ISO as ISO/IEC 14515-1: 2000. This document is also the basis for the current POSIX Conformance Test Suite developed by NIST and used by IEEE to award a "Certificate of Validation" for POSIX conformance. The standard on which the test methods are based, however, was revised in 1996, and is undergoing another revision by the Austin Group. Therefore, since the test-methods standard is based on another standard that is more than 10 years old, it has to be completely out-of-date, right? Well, not really. Certainly, there are out-of-date individual test methods, such as one that looks for a hard-coded POSIX version number, but they are well known and documented by the POSIX test labs. The revision of POSIX.1 in 1996 added more features than it changed, such as the addition of real-time extensions. The test methods standard for POSIX.1 is guilty of lacking test methods for the added features, rather than being obsolete. The bulk of the test methods contained in this standard are still applicable to the current POSIX.1.

The IEEE Standards Board approved an amendment to the test methods for POSIX.1 in March 2000, published as IEEE Std 2003.1b:2000. This amendment is based on the initial real-time extensions defined in P1003.4, which were included in the 1996 version of POSIX.1. Like its predecessor, 2003.1b is guilty of lacking coverage for some real-time extension features added to the 1996 version of POSIX.1. Such will always be the case as long as additions to POSIX evolve faster than the test methods.

POSIX.2 Test Methods

The test methods for the 1992 version of POSIX.2 (Shell and Utilities) are contained in IEEE Standard 2003.2:1996. Like POSIX.1, POSIX.2 has undergone revisions, mostly in the form of added features, while the test methods have remained unchanged. This situation is not likely to change in either the near or the long term, since it is unlikely that anyone will undertake the effort needed to write test methods. The POSIX.2 test methods will be up for reaffirmation ballot by IEEE later this year.

USENIX news

The Felten Case

By the time you read this much will have happened with this affair. Still, we thought you might like to see a copy of the news release sent out on June 6 by the Electronic Frontier Foundation.

The USENIX Association joined the case as a plaintiff since it seems clear that the Digital Millenium Copyright Act (DMCA), as interpreted by the recording industry, threatens to limit what may be presented at USENIX Association conferences and workshops.

Trenton, NJ – The Electronic Frontier Foundation (EFF) today asked a federal court to rule that Princeton University Professor Edward Felten and his research team have a First Amendment right to present their research on digital music access-control technologies at the USENIX Security Conference this August in Washington, DC, despite threats from the recording industry.

When scientists from Princeton University and Rice University tried to publish their findings in April 2001, the recording industry claimed that the 1998 Digital Millennium Copyright Act (DMCA) makes it illegal to discuss or provide technology that might be used to bypass industry controls limiting how consumers can use music they have purchased.

Like most scientists, the researchers want to discuss their findings and publish a scientific paper about the vulnerabilities of several technologies they studied. Open discussion of music customer control technologies has resulted in improved technology and enhanced consumer choice.

"Studying digital access technologies and publishing the research for our colleagues are both fundamental to the progress of science and academic freedom," stated Princeton scientist Edward Felten. "The recording industry's interpretation of the DMCA would make scientific progress on this important topic illegal."

Felten's research team includes Princeton University scientists and plaintiffs Bede Liu, Scott Craver, and Min Wu. Also members of the research team and plaintiffs are Rice University researchers Dan Wallach, Ben Swartzlander, and Adam Stubblefield. Another scientist and plaintiff is Drew Dean, who is employed in the Silicon Valley. The USENIX Association has joined the case as a plaintiff.

The prominent scientist and his research team originally planned to publish the paper in April at the 4th International Information Hiding Workshop. However, the scientists withdrew the paper at the last minute because the Recording Industry Association of America (RIAA) and the Secure Digital Music Initiative (SDMI) Foundation threatened litigation against Felten, his research team, and the relevant universities and conference organizers.

SDMI sponsored the "SDMI Public Challenge" in September 2000, asking Netizens to try to break their favored watermark schemes, designed to control consumer access to digital music. When the scientists' paper about their successful defeat of the watermarks, including one developed by a company called Verance, was accepted for publication, Matt Oppenheim, an officer of both RIAA and SDMI, sent the Princeton professor a letter threatening legal liability if the scientist published his results.

EFF filed the legal challenge in New Jersey federal court against RIAA, SDMI, Verance, and the U.S. Justice Department so that the researchers need not fear prosecution under DMCA for publishing their research.

"When scientists are intimidated from publishing their work, there is a clear First Amendment problem," said EFF's Legal Director Cindy Cohn. "We have

USENIX MEMBER BENEFITS

As a member of the USENIX Association, you receive the following benefits:

FREE SUBSCRIPTION TO *;login:*, the Association's magazine, published eight times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on security, Tcl, Perl, Java, and operating systems, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

ACCESS TO *;login:* online from October 1997 to last month <*www.usenix.org/publications/login/login.html*>.

ACCESS TO PAPERS from the USENIX Conferences online starting with 1993 <*www.usenix.org/publications/library/index.html*>.

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, and election of its directors and officers.

OPTIONAL MEMBERSHIP in SAGE, the System Administrators Guild.

DISCOUNTS on registration fees for all USENIX conferences.

DISCOUNTS on the purchase of proceedings and CD-ROMs from USENIX conferences.

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals. See <*http://www.usenix.org/membership/specialdisc.html*> for details.

FOR MORE INFORMATION REGARDING MEMBERSHIP OR BENEFITS, PLEASE SEE

<*http://www.usenix.org/membership/membership.html*>

OR CONTACT

<*office@usenix.org*>

Phone: +1 510 528 8649

FOR INFORMATION ABOUT CONFERENCES, PLEASE SEE

<*http://www.usenix.org/events/events.html*>

OR CONTACT

<*conference@usenix.org*>

Phone: +1 510 528 8649

long argued that unless properly limited, the anti-distribution provisions of the DMCA would interfere with science. Now they plainly have."

"Mathematics and code are not circumvention devices," explained Jim Tyre, an attorney on the legal team, "so why is the recording industry trying to prevent these researchers from publishing?"

USENIX Executive Director Ellie Young commented, "We cannot stand idly by as USENIX members are prevented from discussing and publishing the results of legitimate research."

EFF is challenging the constitutionality of the anti-distribution provisions of the DMCA as part of its ongoing Campaign for Audiovisual Free Expression (CAFE). The CAFE campaign fights over-reaching intellectual property laws and restrictive technologies that threaten free speech in the digital age. "The recording studios want to control how consumers can use the music they buy. Now they want to control scientists and publishers, to prevent consumers from finding out how to bypass the unpopular controls," said EFF Staff Attorney Robin Gross.

Media professionals have been invited to attend a June 6 press conference and simultaneous teleconference on the Felten case featuring the legal team and Professor Felten.

The legal team includes EFF attorneys Lee Tien, Cindy Cohn, and Robin Gross. Outside lead counsel Gino Scarselli, argued the Junger case where the 6th Circuit Court of Appeals ruled unanimously that computer code is creative expression worthy of First Amendment protection. Also members of the legal team are James Tyre, a technology savvy lawyer from Southern California who co-founded the Censorware Project and wrote an amicus brief in Universal v. Reimerdes, and Joe Liu, a Professor of Law at Boston College. Local counsel in New Jersey are First Amendment specialists Frank Corrado of Rossi, Barry, Corrado, Grassi and Radell, and Grayson Barber, chair of the ACLU-NJ privacy committee.

For more background on Professor Felten and his team's legal challenge:

http://www.eff.org/sc/felten/

Upturns and Otherwise

by Daniel Geer

President, USENIX Board of Directors

<geer@usenix.org>

The best (only good?) part of accumulating years is a chance to see long waves. I've been on this rock long enough to see trees mature, which should just about settle the question about me, and if that doesn't, this will: A former colleague of mine, talking to his grandfather on what turned out to be his grandfather's deathbed, asked him if he was sorry to be going. The grandfather said, "I have only one regret: I won't get to see how it all turns out."

That so sums up the essence of the long wave, I want to repeat myself. The chance to see long waves is a privilege, and of course, it sure as hell beats the alternative. That said, I've seen downturns before and you'll see them again. While anyone can prosper in fat years, only the well-prepared prosper in lean years. Yeah, there is an element of luck in it, but just like Satchel Paige said, "The harder I work, the luckier I get."

It is during a downturn that evolution happens. It is when there is survival pressure that survival of the fittest has meaning. Everyone who is enough of a USENIX member to be reading this is off to a good start and if you want to read that as elitist, please do. For all that any decent person wants to share a measure of their good fortune, survival is more elemental than decency, and it is a form of power. Power is never given, only taken. Each of us, when confronted with a world in which, especially suddenly, there is less to go around, will have to choose what they want to maximize, and I suggest you maximize your marketability.

Speaking purely idiosyncratically, which is to say with the subjective bias of personal myopia, what has worked for me through several complete turns of the business cycle is to always be worth more than I am paid, to never assume that any saleable skill necessarily comes with a durable market, that hybrid vigor works for careers even better than it works for seed corn, that nothing begets loyalty like a frightening commitment to excellence in preference to maneuvered advantage, and that more opportunities exist than anyone can ever exhaust but only on the condition that you keep your eyes open enough to notice the blamed things.

Where does USENIX fit in this? For me, at least, it has been one long tub soak in hybrid vigor, a daily wake-up call telling me what it is I do not know and didn't even know I didn't know, a place to teach and be taught, to be put on the spot in so many ways by people who are the best there is at what they do that I just couldn't help absorb something, by osmosis if nothing else. For much of my career, I have attended USENIX on my own nickel, and I always figured that the pain was more than compensated by the gain. You can invest in yourself any way you damned well please – I'm in no position to tell you how to run your life – but if you want a career that spans more than one business cycle, you had better invest in things that are durable, that make your survivability more probable

than for the guy sitting next to you. For me, memorizing as much as I could of the combined proceedings of USENIX has been, at once, impossible and essential. Building up a web of colleagues who are as good, as pervasive, as central, as insightful, as bizarre as the USENIX attendees has made me resistant if not immune to business cycles.

However much your mileage may vary, check your gas gauge. I recommend you fill up here. Self-serve if you have to.

Update on EFF DMCA Cases

by Cindy Cohn

Legal Director, Electronic Frontier Foundation

Cindy@eff.org

The Electronic Frontier Foundation (EFF) is pursuing several legal cases to protect copyright and fair-use rights by opposing the anti-circumvention rules of the Digital Millennium Copyright Act (DMCA) on the grounds that the act violates the constitutional right to free expression. The cases build on EFF's earlier precedent-setting victory, *Bernstein v. U.S. Department of Justice*, where a federal appeals court ruled that code is free speech and, therefore, protected by the Constitution.

USENIX has generously supported our work on these important cases. In turn, we will attempt to give regular updates to *;login:* so that USENIX members can watch our work as it develops. While this support has been greatly needed and appreciated, the cost of this effort has greatly outstripped our annual budget, even with the support of USENIX. As a member-supported organization, the EFF relies on the backing of those who believe that free speech is essential. If you believe we are doing the right thing in opposing the DMCA, we invite you to join EFF and to assist us in our efforts.

BACKGROUND

The Digital Millennium Copyright Act was introduced in Congress several years before it actually passed in 1998. From its inception, the law was rife with problems for free speech and the growth of technology. Most particularly, the anti-circumvention rules of section 1201 of the DMCA give content holders much broader rights to digital content than they ever held with non-digital content.

Concerned about fair use and reverse engineering, EFF, with several other groups, including members of the library and scientific communities, fought against passage of the DMCA. However, the music, movie and software industries, with their bottomless funding bases, lobbied hard for its passage, and, ultimately, the DMCA became the law of the land.

This law is problematic on several levels. Most importantly, it will eviscerate the public side of the copyright bargain — the part that recognizes that the goal of the copyright monopoly is to give authors the incentive to produce works so that eventually those works will fall into the public domain or be available for fair use or ordinary use to all people. The DMCA effectively eliminates fair use by letting content owners use technology to completely control all uses of their works. This has already come to a head in the *2600* case (see below), where content owners have gone after an electronic newspaper for publishing computer code.

Also troublesome is the criminalization of circumvention software based upon its possible misuse, even though it has plain and important acceptable uses. This has also come to a head in the *2600* case, where software that circumvents the encryption code used on DVDs was posted on the Internet to facilitate the creation of a DVD player using the Linux operating system. The court held that since the software could be used to pirate DVDs, it was in violation of the DMCA.

Finally, the impact on science could be quite severe, since those who seek to do

USENIX BOARD OF DIRECTORS

Communicate directly with the USENIX Board of Directors by writing to <*board@usenix.org*>.

PRESIDENT:
Daniel Geer <*geer@usenix.org*>

VICE PRESIDENT:
Andrew Hume <*andrew@usenix.org*>

SECRETARY:
Michael B. Jones <*mike@usenix.org*>

TREASURER:
Peter Honeyman <*honey@usenix.org*>

DIRECTORS:
John Gilmore <*john@usenix.org*>
Jon "maddog" Hall <*maddog@usenix.org*>
Marshall Kirk McKusick <*kirk@usenix.org*>
Avi Rubin <*avi@usenix.org*>

EXECUTIVE DIRECTOR:
Ellie Young <*ellie@usenix.org*>

encryption research that could be used for circumvention by others must effectively clear their work with the content industry ahead of time or face liability for publishing it. Science rarely works that way, even where the results could affect national defense. We've just recently seen the beginning of this problem, as the SDMI, a record industry coalition, issued threatening letters to Professor Edward Felten and others. The threat succeeded in convincing the professors to withdraw a paper about the breaking of the watermarks on digital music from a scientific conference in late April 2001.

LEGAL CASES

2600 Case

In January of 2000, the movie industry brought a lawsuit under the DMCA. This case was brought in the federal District Court in the Southern District of New York against *2600 Magazine* based upon the discovery that the secret key to the weak encryption system used on DVDs was posted all over the Internet. The eight major motion picture studios sued *2600 Magazine* based upon its publication of the DeCSS code, its news coverage of the controversy, and the large number of links the magazine provided to the code.

From the outset the EFF knew this case was not going to go well; the judge in the case, Judge Kaplan, sided with the industry from the very first hearing. We fought a temporary restraining order, but the court found that the anti-circumvention rules of the DMCA prevented *2600 Magazine* from publishing or even linking to DeCSS, because it could be used to circumvent the encryption placed on DVDs. CSS is designed to prevent copyright infringement, but the court held that publishing DeCSS was illegal even when no infringement had occurred — despite the fact that it was being used for legitimate, even constitutionally protected, purposes — simply because it could be used for infringement.

The case was argued before the Second Circuit Court of Appeals on May 1, 2001. The lower court's ruling basically says that code is not free speech, the *Bernstein* decision notwithstanding. The appellate briefs describe the lower court's decision as "putting the anti-circumvention rules of the DMCA on a collision course with the Constitution." EFF is asking the Second Circuit to prevent this by interpreting the statute consistent with the First Amendment and settled copyright laws.

In late January, eight *amicus* briefs were filed in support of EFF's appeal of the injunction against *2600 Magazine*, including from the ACLU, the Digital Future Coalition, librarians, journalists, computer scientists, law professors, educators, and cryptographers. A sponsor of the computer programmers' brief, noted Princeton University Computer Science Professor Edward Felten, stated, "The lower court's interpretation of the DMCA would effectively shut down research in some areas of computer security by banning the publication of research results in those areas. Ironically, it has already prevented me from publishing research results that could be used to strengthen the protection of copyrighted works."

The EFF successfully convinced noted constitutional scholar and advocate Kathleen Sullivan, dean of the Stanford Law School, to argue the case on behalf of *2600 Magazine*. The Second Circuit will either send the case back down to the trial court for further hearings (if, for instance, we are successful in convincing the appellate court that the trial court misapplied the law), or it will be set for a review of the appellate panel decision by the entire Second Circuit Court and then probably petitioned for decision by the U.S. Supreme Court. While the exact path is difficult to predict, it is likely that this case will continue for at least two to three more years.

More information about this case is available on the EFF Web site at: *<http://www.eff.org/pub/Intellectual_property/Video/MPAA_DVD_cases/>*.

USENIX SUPPORTING MEMBERS

Addison-Wesley
Kit Cosper
Earthlink Network
Edgix
Interhack Corporation
Interliant
Lessing & Partner
Linux Security, Inc.
Lucent Technologies
Microsoft Research
Motorola Australia Software Centre
New Riders Publishing
Nimrod AS
O'Reilly & Associates Inc.
Raytheon Company
Sams Publishing
The SANS Institute
Sendmail, Inc.
Smart Storage, Inc.
Sun Microsystems, Inc.
Sybase, Inc.
Syntax, Inc.
Taos: The Sys Admin Company
TechTarget.com
UUNET Technologies, Inc.

Pavlovich v. DVD-CCA

The DVD Copy Control Association (DVD-CCA), a newly formed association of the Motion Picture Association of America (MPAA), is suing hundreds of individuals who put DeCSS on their Web sites, alleging that the plaintiffs misappropriated trade secrets when they reverse engineered DVD technology. At issue in the case is the First Amendment right to free expression, as well as the right to engage in lawful reverse engineering. EFF is coordinating the defense, representing Andrew Bunner and paying for additional outside counsel for him. Mr. Bunner is the only defendant who has been properly brought into the California courts. The superior court issued an injunction preventing the publication of DeCSS by the defendants, and we have appealed. The First Amendment Project, a California-based nonprofit located in Oakland, has been serving as lead counsel on the appeal.

In addition, EFF co-operating counsel Allon Levy has also represented Matthew Pavlovich, one of the many named defendants who are not properly sued in California. Pavlovich won an initial victory in December 2000 when the California Supreme Court granted his petition for review based on lack of personal jurisdiction and sent the matter back to the appellate court for further review. EFF also assisted in locating counsel for Derek Fawkus, another person who claimed that California jurisdiction was improper since he is based in Scotland and has never even visited California.

Both issues – the jurisdiction question concerning Mr. Pavlovich and the appeal of the preliminary injunction by Mr. Bunner – have been fully briefed and are currently scheduled to be heard together by the California Court of Appeals at a date to be set soon. We expect the arguments to be held in May or June 2001. The remainder of the case has been put on hold by the California Supreme Court pending the appellate court determination.

More information about the case is available at: *<http://www.eff.org/IP/Video/DVDCCA_case/>*.

OTHER RELATED PROJECTS

In addition to the litigation, we are doing ongoing public awareness work around this issue, with speeches, press work, and grassroots efforts under our CAFE project (Campaign for Audiovisual Free Expression). CAFE is a multi-avenue campaign to educate and engage the public in the issues surrounding fair-use rights, and in particular the DVD/DeCSS legal cases. Campaign strategies include public discussions through our BayFF public forum; posting up-to-date information on the DVD/DeCSS cases on our Web site, with extensive links to other sites; media coverage; and most recently, our Radio EFF Program. We have also recently released the "Open Art License," an attempt, with homage to the Free Software Foundation and the open source movement, to allow artists to release their works to the public to be freely copied and used as long as original attribution is made. We believe that unless people outside the technology community understand this issue, we will not be successful in combating the DMCA.

The CAFE project Web page is available at: *<http://www.eff.org/cafe/>*.

CONCLUSION

We recognize that the battle against section 1201 of the DMCA will be a long, difficult one. We expect it to continue for at least five years and to change as the technology to limit access to digital content continues to develop. Although DVDs are currently at the center of the dispute, it is only a matter of time before books, music, multimedia tools and content as diverse as human thought will be similarly locked up and metered out to us. We believe that such a scenario would create a world much different than our current one – one in which we would be much the poorer for the loss of access to what, at the end of the day, is really our shared culture. From the standpoint of technological progress, we also see that criminalizing the software tools that might be used for illegal purposes, even if they are not being used illegally, is likely to become more and more prevalent if 1201 is allowed to stand. The scientific process itself, and the Internet's promise of science freely available to all interested persons, not just those pre-selected to sit in ivory towers or corporate offices, is ultimately at risk.

Happy 20th Birthday FRUUG!

by Steve Gaede

Steve Gaede has been FRUUG coordinator since 1984. In his early years, he created UNIX capacity planning tools. Today he undertakes a variety of research and prototyping projects through his company Lone Eagle Systems Inc.

gaede at loneagle.com

April 1981 – April 2001

April 2001 marked the 20th birthday of the Front Range UNIX Users Group (FRUUG), making it the oldest, still-running local UNIX users group around. Our ripe old age – probably a century in high-tech years – provides a good excuse for a bit of senile reminiscing about all we've managed to accomplish in these two decades. It turns out that we've been surprisingly on top of quite a few technological developments well before their time; and embarrassingly wrong about a few, too.

FRUUG's largest concentration of members is in Boulder, with membership extending along the Front Range of the Rocky Mountains from Pueblo, Colorado, to Cheyenne, Wyoming. The group meets roughly monthly, scheduling meetings around the availability of interesting talks and speakers rather than attempting to meet on a particular day each month. It currently has close to 300 members, with around 70 attending any given meeting.

Though it started as a sort of UNIX support group, it exists today more as a forward-looking computing technology group, not limited to UNIX operating system topics. Despite its changing role, one facet of FRUUG has remained consistent: it has served as a gathering place and a stable touchstone for computing professionals to meet and make contacts for more than two decades.

The Early Years

In 1981, Dick Hackathorn and Rick Patch founded the Boulder Users Group (BUG), named without the adjective describing what it was we *used* because in those days nobody dared toy with the sacred trademark of Bell Laboratories. The group quickly grew beyond the boundaries of Boulder and its members voted to re-name it in early 1982. Those (including the author) who preferred the more colloquial sound of BUG still tend to pronounce FRUUG as if it rhymes with BUG.

In those days, Boulder was a relative hotbed of UNIX activity, with research institutions like the University of Colorado, the National Center for Atmospheric Research (NCAR), and the National Institute of Science and Technology (NIST), as well as commercial organizations like Bell Labs, Cray Labs, NBI, and Storage Technology working with the UNIX operating system. One of the first USENIX conferences was held in Boulder in 1980, pre-dating FRUUG's founding by a year. Though not officially affiliated with any national group, FRUUG's meetings for years included reports on current events from the most recent USENIX conferences.

One of the features that put Boulder on the UNIX map was the fact that the High Altitude Observatory's UNIX machine (hao) was a key component in the UUCP networking backbone that enabled UNIX systems to transfer mail to each other. For those who didn't experience those days, UUCP stands for "UNIX-to-UNIX Copy" and was the basis for a store-and-forward network that was used to copy messages to a remote system (usually over modem connections) and then remotely execute a mail program to send them on to their next hop. The network was completely *ad hoc*. Mail addresses specified the route to be taken, and the whole thing depended on making a lot of personal contacts to establish connections.

The early meetings were small enough that they could be hosted by just about any company that had a few chairs. They usually included a brief talk, a tour of whatever facility we visited, and a round-table discussion that provided a forum for people to ask questions like: "Can I set up a UUCP connection to you," or, "Do you have a driver for such and such a disk?" The unspoken question often on members' minds was: "What would it be like to work here?"

The days in which meetings toured up and down the Front Range gave us a good feel for the UNIX activity in the environs.

The community of those using the UNIX operating system was relatively small and insular, and there was a fair amount of circulation between companies as interesting projects came and went. A memorable fall 1981 meeting was held in an outpost of Interactive Systems that occupied the old Rocky Mountain National Park headquarters in downtown Estes Park; that round-table discussion took place around a roaring fire in a huge stone fireplace with a fall snow beginning outside.

The Dawn of Desktop Computing

The early 1980s saw the convergence of three technical advancements that would form the basis for the desktop computers that we all use today. The Motorola 68010 family of microprocessors provid-

November 11, 1981

...Front Range UNIX* Group

...: John L. Donnelly

SUBJECT: ...Meeting, Thursday, 11/19/81

The next FRUG meeting is set for Thursday, November 19, 1981, 1130-1500 in the Main Seminar Room at NCAR. Membership in the group is widely spread along the Front Range of the Rockies, thus the liberty was taken to address the group as FRUG. Credits to Ken Dumenico of NCAR.

NCAR has a world renowned cafeteria famous for its eclectic delights. It is open 1130-1330 weekdays. You might wish to plan your visit to indulge its gourmet delicacies. No one takes credit/no responsibility for the cafeteria's fare.

The meeting agenda includes an overview of NCAR's use of UNIX in connecting its 11/70s to the various nodes on the NSC hypercha... net. The Software Tools package has also been implemented at NCAR and its uses will be described. COMPUTERLAND has indicated its willingness to display the UNIX based Fortune 32:16 microcomputer at the meeting as well. Finally, a tour of the multi-million dollar NCAR Computing Division can be arranged for those of you who can eloquently elope through the first FRUG meeting held at NCAR.

Map direction you to the hallowed halls of science(NCAR) fol...lows:

Please watch out for the deer, bighorn and miscellaneous necessarily in that order on the NCAR hill.

...the main entrance and the security guard will direct you to ...e Main Seminar Room and/or the cafeteria.

* UNIX is a trademark of Bell Laboratories

Meeting announcement for a 1982 meeting at NCAR; including a review of the cafeteria's "gastric delights."

ed support for memory mapping, enabling the UNIX operating system to run on desktop computers with isolated virtual memory for each process – one of those basic features that the Wintel world wouldn't implement until more than a decade later. Winchester disks became smaller than a filing cabinet and could fit into desktop workstations. And bitmapped displays enabled the graphical user interfaces that brought a new meaning for the word "mouse" into the vernacular.

Demonstration of an early UNIX workstation at NBI in 1985 (from left to right: Steve Gaede, Ron Hughes, Bruce Sanders)

We heard from quite a number of Motorola 68000-based UNIX system vendors as the desktop computing world developed, and many of them have been long forgotten. We had a demonstration of a workstation by Fortune Systems in November 1982. Masscomp showed us multiprocessing based on Motorola processors in March 1984. We heard about UNIX workstations from NBI and Integrated Solutions in 1985. Bill Joy, from an outfit called Sun Microsystems, showed us a system that looked like many others at the time. The Sun 2/120 boasted a Motorola 68010 processor, bitmap display, optical mouse, and of course Berkeley 4.2BSD UNIX with a kernel-based windowing system. Who would have guessed how the landscape would change between then and now.

Although many of us hoped that the 68000 series would win the microprocessor cook-off by virtue of its clean design, the UNIX community didn't ignore Intel architecture processors. The IBM PC came on the market, and it didn't take long for the UNIX operating system to be ported to Intel 8086 processor-based machines even without memory mapping support. When Intel released the 286 processor, UNIX was ported to it before DOS was, and systems and software were available from AT&T, Microport, and Xenix. We've had meetings through the years on UNIX for the PC, including talks from the folks at BSDi, from Bob Gray and Dick Dunn on "Cheap UNIX," and meetings on Linux as it arrived on the scene.

In the early 1980s, window systems were typically kernel-based, but in 1986 we hit it right with a talk on the X Window System. Despite how Sun came out ahead of all of the other workstation vendors we heard from, our Sun-sponsored talk on NeWS (Network Extensible Window System) was one of those innovations that didn't get very far. We still have (somewhere) a video of the Great X Windows Debate that pitted the X Window System against Sun's NeWS, Microsoft Windows, and Apple QuickDraw in February 1988.

Network computing became a hot topic in the late 1980s. We heard about Integrated Solutions' Transparent Remote File System (TRFS), Apollo's Network Computing System (NCS), and of course Sun's Network File System (NFS). For remote procedure calls, the debate raged between Open Network Computing (ONC) and Distributed Computing Environment (DCE).

The C programming language encountered some competition from its object-oriented cousin C++, on which we hosted our first meeting in 1988. Many related topics, like the Standard Template Library and Design Patterns, followed as the years went on.

The Internet Appears on the Radar Screen

Our 10th Anniversary FRUUG meeting announcement in April 1991 was a double issue on real paper as we introduced Colorado SuperNet (CSN) to FRUUG members, with the first, local, commercial offering of dialup UUCP and SLIP services. Colorado SuperNet was one of the first Internet service providers anywhere, receiving state funding to promote the use of the Internet within Colorado for research, education, and – for the first time – business. We promoted the nonprofit, state-funded CSN for a number of newsletter issues, helping our members become aware of this great alternative to ad hoc UUCP connectivity and the long-distance dialup services provided by UUNET. CSN was eventually groomed for a corporate takeover. Qwest did the deed, and then shut them down over the holidays just this year – bringing some finality to our tax-funded efforts.

Though at the time they were years away from becoming FRUUG Executive Committee members, Neal McBurnett and Joe VanAndel demonstrated tools for surfing the Internet in February 1994, including such classics as Mosaic and Lynx. Remember Lynx? In our 1994 meeting announcement we touted it as the less "resource-intensive" alternative to Mosaic, suggesting that the systems we used weren't quite as powerful as they are today. That February meeting was followed by an ISP cook-off with the Colorado-based ISPs presenting — and debating – their various benefits. Internet pioneer Mike O'Dell kicked off the meeting with a presentation on how the Internet worked at the time, and we had nearly a full house in the 500-seat NIST auditorium, the largest facility available to us.

Keeping Us Up-to-Date

The middle and late 1990s saw a continuation of our trend of keeping members

up-to-date on emerging technologies including networking, window systems, security, Internet connectivity, software engineering, and programming languages. The acronyms describing some of our most recent five years of meetings are sometimes dizzying: Y2K, PDA, STL, RPC, ONC, DCE, ISDN, ADSL, MPEG, DNS, BIND, XML, RMI, AWT, and

The FRUUG Executive Committee planning the next meeting, from a 1996 Boulder Daily Camera *article. (from left to right: Tom Cargill, Steve Gaede, Wally Wedel, Carol Meier, Mark Carlson)*

JDBC, OMG, CORBA, and even NT, COM, and OLE. Which of these will be forgotten in a decade?

We had Perl tutorials from Tom Christiansen and even an appearance by Larry Wall. John Ousterhout gave us his vision for Tcl/Tk and his concept of agents. We heard about open source from Richard Stallman, about cyberterrorism from Rob Kolstad, and the potential horrors of Y2K from Evi Nemeth. In 1995 James Gosling visited us to talk about his browser called HOTJAVA, and as a side note discussed the features of the experimental programming language called Java used to create it.

An Interesting Trip

In November 1999 Bill Joy chatted with us about his journey "from BSD to Jini," and shared quite a few interesting stories about the technology that has been developed during those years. It's been an interesting trip for FRUUG as well, and we hope that the next decade will bring us as interesting a time as the last two have been — and with the continued involvement of our FRUUG members, it no doubt will be.

An Invitation

We invite you to visit the FRUUG Web site at *http://www.fruug.org* and peruse some of the relics from our meeting archive. We're working on getting as many of the old artifacts online as possible.

Thanks!

Thanks to all those who have contributed to FRUUG over the years, especially the FRUUG Executive Committee for contributing to this historical perspective and gathering for monthly lunches to discuss technical topics of the day – and plan meetings. The FRUUG Executive Committee currently includes: Tom Cargill, Mark Carlson, Barb Dijker, Dick Dunn, Steve Gaede, Neal McBurnett, Carol Meier, Bill Meine, Joe VanAndel, and Wally Wedel.

25 Years Ago

by Peter H. Salus
USENIX Historian
<peter@matrix.net>

In the depths of my mouldering masses of paper lies a copy of a letter from Lew Law ("Director of Technical Services," Harvard Science Center) to Mel Ferentz, dated June 24, 1976.

It begins:

"The Science Center is upgrading its present computer system which runs UNIX from an 11/45 to an 11/70. As a result we wish to sell the following:

(1) 11/45 CPU with memory management KY11C (serial number 1147)
(2) Hardware bootstrap
(3) KW11L – line frequency clock
(4) DL11 – single asynchronous serial line interface
(5) 24K non-parity DEC core
(6) FP11B floating point processor $35,800
(7) 96K non-parity core – Cambridge Memory Expandacore 11 $13,200
(8) RS04 controller only for fast swapping disc $5,400

Items 1 to 6 are to be sold as a package. Items 7 and 8 could be sold separately. All DEC equipment was purchased 7/1/74 and has been under maintenance contract since installation . . . ."

Wow!

This really illustrates Moore's law to me. 96K core memory for $13,200! 25 years later, I can buy 45G for under $150 retail.

Lew also supervised the publication of the UNIX manuals – in 1976 this was the justly famed sixth edition. He wrote:

"Printing and shipping of the UNIX documents seems to have gone quite well – we have ordered over 200 Programmers Manuals and 170 Documents. Most of these have already been shipped."

There was an order form in *UNIX NEWS*, 6.

That issue of *UNIX NEWS* also informed us of the move of Western Electric's Patent Licensing office to Greensboro, NC. Richard G. Shahpazian, "Patent License Manager."

Best Papers Online!

Over the past decade, the Program Committees from many of the USENIX conferences and workshops have given out Best Paper, Best Student Paper, and Best Presentation awards. A list of these awards, with links to the actual papers is now available at *http://www.usenix.org/publications/library/proceedings/best_papers.html* Note: You do not need to be a USENIX member to access the papers in this compendium.

Announcement and Call for Papers **USENIX**

BSDCon 2002

The premiere conference for the BSD community

February 11-14, 2002, Cathedral Hill Hotel, San Francisco, California
http://www.usenix.org/events/bsdcon02

Important Dates

Refereed Paper abstracts due: *August 27, 2001*
Invited Talk proposals due: *August 27, 2001*
Notification to authors: *October 1, 2001*
Final papers due: *December 4, 2001*

Conference Organizers

Program Chair
Sam Leffler, *Errno Consulting*
Program Committee
Chris G. Demetriou, *Broadcom Corp.*
Jun-ichiro itojun Hagino, *IIJ Research Laboratory/KAME Project*
Jordan K. Hubbard, *Wind River Systems*
Rob Kolstad, *Delos*
Perry E. Metzger, *Wasabi Systems, Inc.*
Jim Mock, *O|S|D|N*
Ernest N. Prabhakar, *Apple*
Gregory Neil Shapiro, *Sendmail, Inc.*

Overview

The Berkeley Software Distributions (BSDs) represent one of the oldest and most vigorous streams of Open Source development. Together, OpenBSD, FreeBSD, NetBSD, Darwin, and BSD/OS represent millions of servers and desktops. The BSDs have long been part of the backbone of the Internet, in everything from embedded applications to large server installations, and will soon be widely deployed on consumer desktops. If you want to develop cutting-edge network applications, then BSDCon is the place to be. Meet all the movers and shakers of the BSD community, and learn how you can use BSD as part of your enterprise-grade solutions.

This is the third BSDCon, but the first to be sponsored by the USENIX Association. Two days of tutorials will precede two days of technical sessions and a vendor exhibit. The combination of technical tracks, invited talks, tutorials, Birds-of-a-Feather sessions, and Work-in-Progress reports provides an opportunity for people of all experience levels to learn from BSD experts, professionals with real world experience, and industry leaders.

Technical Sessions, February 13–14, 2002

Two days of technical sessions feature refereed papers and invited talks by community experts and leaders. Refereed papers are from the community and can win valuable cash and prizes. Papers are published in the Proceedings which are provided to all conference attendees. Refereed papers present problems and solutions in all areas, from kernel internals to real world practical experience.

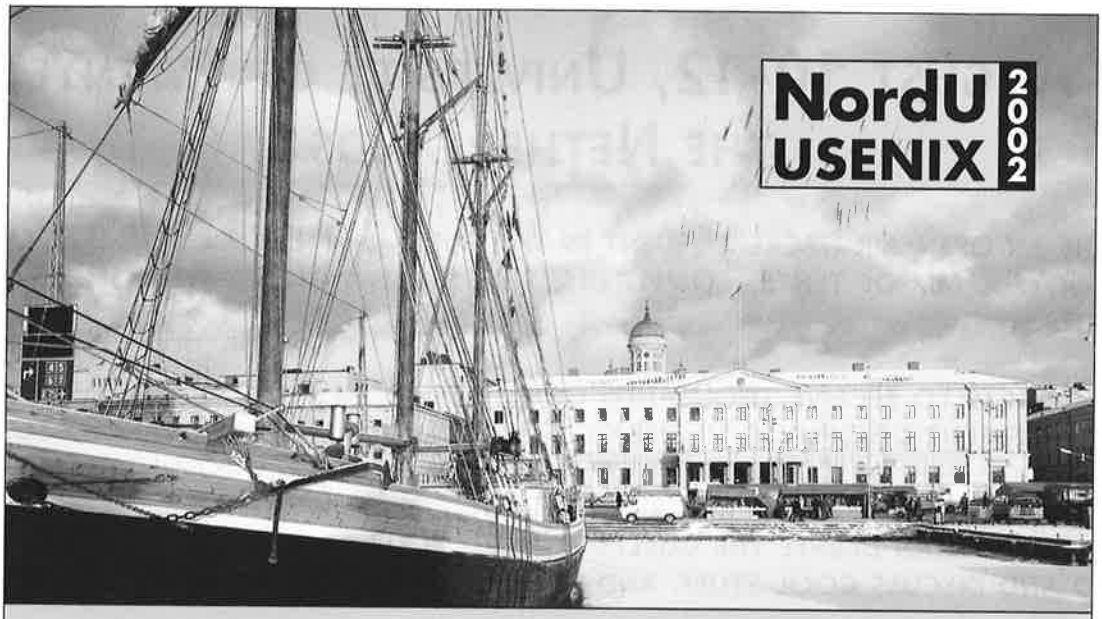
BSDCon seeks refereed papers on topics related to BSD-derived systems and the Open Source world. Topics of interest include:

- Embedded BSD application development and deployment
- Real world experiences using BSD systems
- Comparison with non-BSD operating systems; technical, practical, licensing (GPL vs. BSD)
- Tracking open source development on non-BSD systems
- BSD on the desktop
- I/O subsystem and device driver development
- SMP and kernel threads
- Kernel enhancements
- Internet and networking services
- Security
- Performance tuning
- System administration

Selection will be based on the quality of the written submission and whether the work is of interest to the community. Please see the detailed author guidelines on the web site, including sample extended abstracts and final papers: *http://www.usenix.org/events/bsdcon02/cfp/guidelines.html.*

Program and Registration Information

Complete program and registration information will be available in November 2001 at the Conference Web site at *http://www.usenix.org/events/bsdcon02*. The information will be available in both html and a printable PDF file.



Announcement and Call for Papers

NordU2002 - The fourth NordU/USENIX Conference
February 18–22, 2002, Helsinki, Finland

Information regarding The fourth Nordic EurOpen/USENIX Conference, to be held in Helsinki, Finland, February 18-22, 2002.

A conference organized by EurOpen.SE - The Swedish Association of Unix Users, and an affiliate of USENIX, the Advanced Computing Systems Association.

Important Date
Extended abstracts due September 7, 2001
Notification of acceptance October 12, 2001
Final papers due December 7, 2001

Authors are invited to submit a 1/1 page abstract in English on any of the topics below to the Congress Secretariat. Submission should be original work and will be reviewed by the Technical Review Committee.

All accepted papers will be available on Internet after the Conference. Authors must register for the conference and present their papers in person. Instructions for preparing final papers will be sent with the letter of acceptance.

Complete programme and registration information will be available by mid October 2001. To receive information about the fourth NordU/USENIX Conference, please visit **http://www.nordu.org/NordU2002/** or send an e-mail to **NordU2002@europen.se**

TOPICS
- Security
- Operating Systems
- Open Source/Free Unix
- Software Development
- Interoperability
- High Availibility

Technical Review Committee:

Please visit http://www.nordu.org/NordU2002/

Please send your abstracts to:
Congrex Sweden AB
Attn: NordU2002
P.O. Box 5619, SE- 114 86 Stockholm
SWEDEN
Phone: + 46 8 459 66 00 Fax: + 46 8 661 91 25
E-mail: congrex@congrex.se

.SE EurOpen

USENIX

DKUUG EurOpen

NUUG

FUUG

Hackers at Large 2001

August 10-12, University of Twente, the Netherlands

HAL 2001 is an open-air hackers' event in the spirit of HEU'93, HIP'97, and CCC'99: a joyful mix of tents, computers, sun, presentations, workshops, and social opportunities.

HAL 2001 expects to receive thousands of guests from all over the world and from many different disciplines to debate issues ranging from advanced technical issues regarding some obscure aspect of the Internet to easy-to-understand lectures on some of the dangers of the information society, as well as many, many other topics. But more than debate the guests at HAL 2001 take ample time to get online, relax, build and discuss cool stuff, and engage in good old analog interfacing.

HAL 2001 is for those that can truly celebrate the Internet and embrace new technologies, without forgetting their responsibility to tell others that new technologies come with new risks to the individual and society as a whole.

Topics at HAL include the following:

- privacy & computer security;
- "hacks" – closing the gap between first generation hackers and the younger generation;
- biometrics;
- weird science – creative uses of technology;
- the mutual construction of an Internet nation state.

For sponsorship inquiries, proposals for workshops, subevents, and presentations, please contact us at *grit@hal2001.org*.

You can find more information on our website at: *http://www.hal2001.org*

We hope to see you in August!

The HAL 2001 organisation

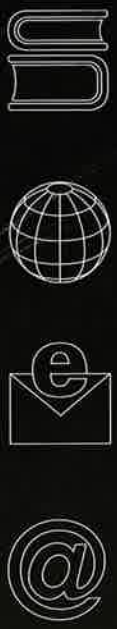
The Internet was invented so millions of people around the world could share information.

We can fix that.

INTERHACK

AT INTERHACK, we stop malicious hackers. We do it with continuous research that finds the "holes" before they do. We do it with customized, integrated solutions. With the most sophisticated programming this side of the Defense Department. And with a perverse sense of satisfaction. So if you have certain information you want to keep to yourself, log on to web.interhack.com or call +1 614 332 4232.

CONNECT WITH USENIX & SAGE



MEMBERSHIP, PUBLICATIONS, AND CONFERENCES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: +1 510 528 8649
FAX: +1 510 548 5738
Email: <office@usenix.org>
<login@usenix.org>
<conference@usenix.org>

WEB SITES

<http://www.usenix.org>

<http://www.sage.org>

EMAIL

<login@usenix.org>

COMMENTS? SUGGESTIONS?

Send email to <ah@usenix.org>

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <*login@usenix.org*> or through the postal system to the Association office.

The Association reserves the right to edit submitted material. Any reproduction of this magazine in part or in its entirety requires the permission of the Association and the author(s).

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send address changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES