;login:

THE MAGAZINE OF USENIX & SAGE

April 2001 • volume 26 • number 2

inside:

CONFERENCE REPORTS

14th Systems Administration Conference (LISA 2000)

A Report from COOTS 6 by Peter Salus

COMPUTING

Needles in the Craystack: When Machines Get Sick, Part 3

OPEN SOURCE

A Logging and Tracing Facility for an Embedded Source Code UNIX Product

PROGRAMMING

Using New Types and Values in C9X

The Tclsh Spot

and much more . . .

SYSADMIN

Introducing Peep: The Network Auralizer

ISPadmin

THE WORKPLACE

You've Been Cracked . . . and Now You're Sued!

Do You Have Both ORs in the Water?

USENIX & SAGE

The Advanced Computing Systems Association & The System Administrators Guild

USENIX Upcoming Events

2001 USENIX Annual Technical Conference

JUNE 25–30, 2001
Boston, Massachusetts, USA

http://www.usenix.org/events/usenix01

Camera-ready final papers due May 1, 2001

The Systems and Network Administration Conference (SNAC)

Sponsored by USENIX & SAGE

JULY 30–AUGUST 3, 2001
Dallas, Texas, USA

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

10th USENIX Security Symposium

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

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

Camera-ready final papers due May 2, 2001

5th Annual Linux Showcase and Conference

NOVEMBER 6–10, 2001
Oakland, California, USA

Refereed Paper abstracts due: June 5, 2001

Invited Talk Proposals due: June 5, 2001

Notification of Acceptance: July 23, 2001

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

Extended abstracts due: June 5, 2001

Notification of acceptance: July 2001

Camera-ready final papers due: October 1, 2001

First Conference on File and Storage Technologies (FAST)

JANUARY 28–29, 2002
Monterey, California, USA

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

Submissions due: July 13, 2001

Notification to authors: September 14, 2001

Camera-ready final papers due: November 15, 2001

2002 USENIX Annual Technical Conference

JUNE 9–14, 2002
Monterey, California, USA

16th Systems Administration Conference (LISA 2002)

Sponsored by USENIX & SAGE

NOVEMBER 3–8, 2002
Philadelphia, Pennsylvania, USA

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

contents

2 MOTD *by Rob Kolstad*

3 APROPOS *by Tina Darmohray*

CONFERENCE REPORTS

4 14th Systems Administration Conference (LISA 2000)

28 COOTS 6 *by Peter Salus*

COMPUTING

30 Needles in the Craystack: When Machines Get Sick, Part 3 *by Mark Burgess*

OPEN SOURCE

37 A Logging and Tracing Facility for an Embedded Source Code UNIX Product *by Bob Gray*

PROGRAMMING

42 Java Performance *by Glen McCluskey*

47 Using New Types and Values in C9X *by Glen McCluskey*

51 The Tclsh Spot *by Clif Flynt*

SYSADMIN

58 Introducing Peep: The Network Auralizer *by Michael Gilfix*

62 Musings *by Rik Farrow*

66 ISPadmin *by Robert Haskins*

THE WORKPLACE

71 Do You Have Both ORs in the Water? *by Steve Johnson and Dusty White*

73 You've Been Cracked . . . and Now You're Sued *by John Nicholson*

USENIX FUNDED PROJECTS

77 Mobile Hoarding and Dynamic Grouping *by Ahmed Amer*

BOOK REVIEWS

80 The Bookworm *by Peter H. Salus*

81 DESIGNING STORAGE AREA NETWORKS by Tom Clark *Reviewed by Steve Reames*

82 PC HARDWARE IN A NUTSHELL by Robert Bruce Thompson and Barbara Fritchman Thompson *Reviewed by Rik Farrow*

STANDARDS REPORTS

83 Some Standardization News *by Keld Simonsen*

SAGE NEWS

85 SAGE Elections

85 SAGE Certification Project Update *by Lois Bennett*

USENIX NEWS

88 Good Works *by Daniel Geer*

89 Board Meeting Summary *by Gale Berkowitz and Ellie Young*

90 Twenty Years Ago in USENIX and in UNIX *by Peter Salus*

91 USACO News *by Rob Kolstad*

ANNOUNCEMENTS AND PROGRAMS

94 USENIX Annual Technical Conference

96 5th Annual Linux Showcase and Conference

;login: Vol. 26 #2, April 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.

Cover photo: The welcoming committee at the LISA 2000 reception.

motd

Volunteering

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 was really surprised recently to hear someone speak of volunteering his time as a "two hour/month effort, max." He seemed to feel that his life was so full of activity that volunteering was way down on his list – and on everyone's list, according to him. His career as a system administrator (presumably along with other, unspoken commitments) was consuming all of his time.

I have taken it upon myself to ask my friends and acquaintances about this notion that "no one volunteers any more, they are too busy." Only a few agree with my correspondent, even though he felt that his views were relatively universal.

I have a different philosophy from the non-volunteer. It must be noted right away, though, that I lack the large time commitments of a wife and family. And, for that matter, a full-time job, though that has mostly changed the amount of time that various tasks get rather than the number of tasks.

I think it's important to support science fairs (a typical judge, for example, has a four hour commitment to our science fair, plus transportation time), programming contests, publishing endeavors about caves, and various sundry "little" activities that I personally enjoy. I also think one should occasionally contribute toward one's profession, whether it's creating (or even just releasing) a software tool for the community, writing a paper, reviewing papers or books, writing articles, organizing a panel at a conference, that sort of thing.

While not an overwhelming feeling, I live in fear that some new societal norm will emerge while I, living in the hinterlands, don't get to learn about it until late in the game. I hope this fellow's comments on volunteering aren't true!

That being said, the USENIX volunteer recruitment effort is going well. Many openings still exist, especially to edit this summer's "Special Issue" of *;login:*.

I hope you will consider sharing your time and expertise occasionally. I think volunteering for various projects, whether trail maintenance for your local bicycle highways or contributing to your profession, is one of the things that really enriches both the society of the world at large and the people who volunteer.

;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
Annelise Zamula

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*>
<*conference@usenix.org*>
<*login@usenix.org*>
WWW: <*http://www.usenix.org*>

apropos

More Than Money

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>

We all work for something. I tell my kids that school is their "job" and their grades are their "pay." For adults, it's more concrete; we "do our time, we get our money." For consultants it can be even clearer that the paycheck is tied to the "deliverable." But occasionally we're reminded that there are rewards other than money for a job well done.

Last week I was talking to a colleague of mine and we got started on an old client of ours. We frequently discuss this particular client because, frankly, we didn't like the ending we had there. I guess it's the nature of conscientious people to dwell on regrets over successes. I say "regrets" rather than "failures" here because, realistically, we didn't come close to failing this client, yet my colleague and I just feel that there was so much more we could have accomplished there.

This was a young business with young employees and gobs and gobs of networks and machines, and it was still growing phenomenally. For a network addict it qualified as a "big fix," and for a seasoned admin it called out for vision and leadership; in short it had "all the right stuff" to be a really great network with fabulous challenge and potential to work on.

We jumped right in, quickly getting our bearings and assessing what needed to be done. The application was very demanding and the network growth necessary to keep up with the growing success of the application was nothing short of staggering. We needed to sustain existing capability and simultaneously get a handle on growth, so that machines and networks could be deployed and administered in a predictable and reliable way.

We laid out a course of action, including specific recommendations and action items to bring this test-bed environment up to industry standards for high-profile, highly-reliable, production networks. This is where we got stopped in our tracks. We had assumed that the sub-par condition and design of the network were a result of the extreme pressure to build a network and develop a product in ultra-compressed "Internet" time, coupled with a young and inexperienced system administration staff that, in their defense, probably had never seen a "production network." When we presented them with concrete steps which would ultimately lead to a securable and scalable production network, we discovered a corporate culture which consistently took network security and reliability risks.

Ultimately we had to settle for chipping away at the problem by implementing little pieces of "good practice" wherever we could. But it did not approach the deliberate, top-down plan we felt was warranted. This is the source of our regret and our conversations about "what could have been" and what else we might have done to persuade them to change their ways.

Unlike many of the conversations about this client, this one had been sparked by some unexpected good news. My colleague had gotten mail from one of the client's system administrators who had moved to another employer, saying that he was deploying many of the individual pieces and, most importantly, the overall approach to his new network with great success, and he wanted to thank him for his example and for what he'd learned.

Making a positive impact on the career of another system administrator has got to be one of the most rewarding things to do and worth far more than any paycheck. Sometimes you're fortunate enough to hear about it. I know that single email message will change the tone of every future conversation we have about that client since we'll now have to count it among our major success stories.

conference reports

OUR THANKS TO THE SUMMARIZERS:

Lee Amatangelo
Brian Baggett
Dave Bianchi
Paul Federighi
Jim Flanagan
Steve Hanson
Dave Homoney
Tony Katz
Brendan Kelliher
Brian Kirouac
Vinod Kutty
Eric Lakin
Dave McFerren
John Ouellette
Socrates Pichardo
Nicholas Schrieber
Sam Shaffer
Josh Simon
Doug Stewart
Theo van Dinter
Craig Vershon
Steve Wormley

Photographs taken at LISA 2000 can be found at

<http://www.usenix.org/publications/library/proceedings/lisa2000/confpix/pixindex.html>

14th Systems Administration Conference (LISA 2000)

New Orleans, Louisiana December 3–8, 2000

ANNOUNCEMENTS

Summarized by Josh Simon

The first session started with the traditional announcements from the program chairs, Phil Scarr and Rémy Evard. Phil began with the following:

- There are over 1800 attendees, making this our biggest LISA conference ever.
- There were 85 papers submitted; we accepted 36 (42%).
- The Proceedings ran 378 pages.
- Over 1,800 messages were sent to and from the program chairs to plan the conference.
- Sixty-three percent of the program committee changed jobs between LISA 1999 and LISA 2000.

Thanks go to the program committee, invited talks chairs, network track coordinators, security track coordinators, guru coordinator, WIPs coordinator, the paper readers, Ellie Young and the USENIX staff, Rob Kolstad for his work in typesetting the Proceedings, Lynda McGinley for the terminal room, and all the vendors mentioned in the conference directory.

Rémy Evard then announced the best paper awards:

System Honorable Mention—"Deployme: Tellme's Package Management and Deployment," by Kyle Oppenheim and Patrick McCormick of Tellme Networks.

Best Papers (tie)—"Peep (The Network Auralizer): Monitoring Your Network with Sound," by Michael Gilfix and Alva Couch of Tufts University; and "Tracing Anonymous Packets to Their Approximate Source," by Hal Burch at Carnegie-Mellon University and Bill Cheswick of Lumeta Corporation. Both papers had a student co-author.

Dan Geer, USENIX president, announced the proposed split of USENIX and SAGE, covered elsewhere in this issue. Barb Dijker, SAGE president, echoed Dan's announcement, reiterated the desire for member feedback, and announced the 2000 SAGE Award for outstanding achievement: Celeste Stokely, for her work in collecting and distributing systems administration information for over 10 years.

Keynote Address

THE WORLDWIDE SYNDICATE

J.D. "Illiad" Frazer, User Friendly

Summarized by Jim Flanagan

The artist behind the immensely popular Web cartoon strip "User Friendly" (<*http://www.userfriendly.com*>) related how he ventured into self-syndicating his work, and what he has learned about the traditional syndication model along the way. "User Friendly" recently celebrated its third anniversary and has experienced explosive growth in Web page impressions.

Cartoonists are curious about human nature, and the traditional syndication process causes these people no end of pain by imposing on them a fundamental separation from their audience, which is treated as a market rather than a community. Each layer between cartoonist and audience removes revenue along with artistic control over content. J.D. told how Berke Brethed stopped doing his popular comic, "Bloom County," not because he ran out of material or burned out, but because some of the content offended Donald Trump, who in turn bought the syndicate and sat on the strip.

The syndicates can only do what they do because they control access to the market. The Web, notorious for its lack of control, allows cartoonists to "syndicate" their own work, retain creative control, and be more responsive to current

events, free from the cumbersome review process. Not that all cartoonists need do is sporadically draw cartoons and post them on the Web. They have to be responsible for all of those things that the syndicate provides: creative and legal support and a revenue model.

By keeping artistic control the cartoonist gains direct access to an audience – a community rather than a market – and lots of interesting feedback. But this community also needs to be nurtured, lest things get out of control. For example, an April Fool's joke in which "User Friendly" claimed to have received a cease-and-desist letter from "a large, unnamed software company" (which was reported as fact on Slashdot by co-conspirator Rob "Cmdr. Taco" Malda) resulted in several higher-ups at Microsoft asking J.D. who in Microsoft had sent the letter. Apparently, the readership had simply assumed Microsoft was the culprit and flooded the phones with complaints, threats, and pledges to attack the company's IT infrastructure on Illiad's signal.

The audience also learned a little bit about what happens behind the scenes at "User Friendly," in that there is a largish organization which provides creative assistance, fiscal management, and other overhead tasks, forever dispelling my image of J.D. penning the strips alone in his basement at night and uploading them to the Web.

Questioners from the audience wanted to know if the UF crew had any plans to help other cartoonists get off the ground with self-syndication, and if UF had any intentions of becoming a syndicate themselves. UF is waiting for the right business model before they start involving other artists, and they would never themselves become a syndicate because it makes little sense to start placing layers between the artist and the audience. There are other Web-based cartoonist collectives getting started as well. Finally, we learned that J.D. also likes the Web cartoons "Goats," by Jonathan Rosenberg (<*http://www.goats.com*>), "Sluggy," by Pete Abrams (<*http://www.sluggy.com*>), and "SinFest," by Tatsuya Isheda (<*http://www.sinfest.net*>).

Invited Talks

How Not to Get Fleeced with Employee Stock Options

Jon Rochlis, The Rochlis Group, Inc.

Summarized by Theo van Dinter

Rochlis is not a professional financial planner, but he has a strong interest in the topic and is expecting a degree in financial planning by the end of 2000. The slides from LISA, and a good amount of other option-related information, can be found at <*http://www.rochlis.com/options/*>.

This talk covered a lot of ground: What is stock? What are stock options? What are the tax rules? Is a given stock option offer any good? When should the options be exercised? Unfortunately, Rochlis only got to cover about half of the information in the slides before running out of time. There is a lot of good information, including financial planning-related links, available on the Web site listed above.

The overall conclusions from the talk were:

- Try to get incentive stock options (ISOs) whenever possible.
- Stock option value is a percentage of company market capitalization, not directly related to the number of shares.
- Don't be emotionally attached to options/shares. Do research and sell if it looks like the stock will lose value. Shares can always be bought again at a lower price.
- Be aggressive with tax deductions. Don't lie, but claim everything possible.

SAGE Update

Barb Dijker, SAGE President

Summarized by Lee Amatangelo

Dijker provided a status report on SAGE for the past year, projects in progress, and future plans.

The largest topic looming over the entire LISA 2000 Conference, starting with a statement during the Opening Session from USENIX President Dan Geer, is that talks are underway to determine if the best course of action for both USENIX and SAGE is to have SAGE become a separate entity.

Perhaps the second biggest issue discussed during this update and at other SAGE meetings during the conference was SAGE's Certification Project undertaken as a way to elevate system/network administration to the status of a profession. But getting there is going to take time. Along those lines, SAGE has already done a fair amount toward defining the various levels of system/network/database administration work.

Several student internship programs and interns present at the LISA 2000 Conferences were recognized. The SAGE board is very much in favor of supporting and promoting more internships, and the audience was encouraged to create new internship programs for system/network administrators.

There was also a call put out for mentors. SAGE encourages additional senior members of our profession to help others grow in this profession in the spirit of the whole open source mentality. Those interested in mentoring or being mentored were asked to contact SAGE.

Ideas were presented and solicited on how SAGE can market itself. Marketing will now become even more important if SAGE is to become a separate entity.

Other major highpoints of the update included the SAGE salary survey results. The results are displayed in many ways

based on various criteria. Salaries can be looked at based on gender, years of experience, geographical location, educational level, certifications achieved, and number of systems (and operating systems) supported, to name the biggies.

Following the formal update presentation, the floor was opened for questions. Collective Technologies' Jeff Tyler said that he had not heard a compelling reason from either side as to whether SAGE should stay with USENIX or break off as a separate entity. One response from the board and other members of the audience was that SAGE and USENIX have two very different charters. However, regardless of whether the split happens or not, the board members of both USENIX and SAGE assured everyone that there would still be ventures and events put on jointly by USENIX and SAGE and that a great working relationship between the two groups would be maintained.

The Digital House

Lorette Cheswick

Summarized by Lee Amatangelo

Lorette Cheswick, along with her not-so-famous husband, Bill (I think he may have worked at Bell Labs once), and a little help from their two children, Kestrel and Terence, presented their "Digital House," was very interesting, entertaining, and highly educational talk.

The Cheswicks have a talking house. Whenever the doorbell rings, a message goes out over the intercom system stating, "Someone is at the door." This message is also heard by the visitor and is particularly amusing to younger children.

Other ideas soon followed. Wouldn't it be nice to know when the mail had arrived? So a wireless motion detector is positioned in the back of the curbside mailbox. Whenever the mailbox door opens and shuts, a message goes out over the intercom system, "Honey, you've got mail!" (which was quickly changed due to its political incorrectness).

The intercom system also announces daily astronomical events such as the times for sunrise, sunset, iridium flares, comets, planet risings and settings, etc. In addition, the intercom provides stock quotes throughout the day.

Here are more details from their presentation:

The Cheswick digital house includes:

- Home Ethernet and wireless
- Platform for firewall-free experiments
- Caller ID to serial port
- X10, Linux sound card to home intercom

The Cheswicks have 11 computers on their home LAN right now.

The full Powerpoint presentation is located at: <*http://www.cheswick.com/*>, then select "Powerpoint slides for Lorette's *Digital House* presentation at the New Orleans LISA 2000." Or simply select <*http://www.cheswick.com/ppt/digital.ppt*>.

Experiences with Incident Response at Ohio State University

Steve Romig, OSU-IRT

Summarized by Brendan Kelliher

Before the foundation of the Incident Response Team at Ohio State University, OSU's computer security was conducted on an ad-hoc, part-time basis. In early 1996 Steve Romig, a campus system administrator, was asked to be part of a full time, professional systems security force which would become the OSU Incident Response Team.

In the summer of 1996 the OSU-IRT responded to a local ISP's complaints about attacks emanating from campus systems. A local hacker group, the LoTecs, was illegally accessing university systems and using them as launch points to attack other sites.

Steve traced the hackers back to the campus modem pool. He had to work with the local phone company (AmeriTech), which caused him much aggravation. When tracing a call in real time, he would be passed from operator to engineer, each trying to complete a separate part of the trace.

Some of the hacker's login sessions lasted for days. One lesson he learned is that phone traces can be done "after the fact." The phone company has logs of all calls and can do traces going back for months. This freed him up to concentrate on what systems and accounts the intruders were compromising.

By watching the logins of known hacked accounts, he noticed a pattern emerge in their sign-on/sign-off times. Just after the local schools let out he would see the first logins. Then around dinnertime he would see them logout for a brief period, then sign back on shortly after. His assumption that the intruders were local high school kids would later turn out to be correct.

Steve attended hacker meetings along with a local undercover police officer. They learned the identities of several of the hacker group members, one of whom had a history of traffic violations.

In October of 1996 the hackers started to attack ".mil" sites, which got the attention of the FBI. Steve's information was used to build the government case. One interesting technique the hackers used was to personally respond to complaints from a victim they had hacked. In September of 1997, nine members of the LoTecs were served with warrants.

Here are some recommended steps you can take to guard against breaches in security:

- Determine what measures your company is willing to take against hackers.
- Create an Incident Response Team.
- Log everything!
- Make contact with your local authorities; many police departments have established "white collar" crime units which also cover computer crimes.
- Work out security procedures and emergency contacts with your telco or Internet access provider.
- Never audit your own security.
- Be patient when dealing with LEOs (law enforcement officers).
- Document all your steps when tracking an intruder; your notes will be crucial if the case ever goes to court.

Integrating LDAP into a Heterogeneous Environment

Leif Hedstrom, Netscape

Summarized by Jim Flanagan

Hedstrom started this talk with a brief overview of what LDAP is and is not. LDAP is a client-server protocol with its roots in the OSI directory standard X.500. LDAP is not a database but merely the protocol for transmitting and formatting directory services information on the network. An LDAP entry consists of one or more attribute-value pairs, and can have multiple values associated with a single attribute. The objectclass attribute defines what attributes an entry may legally have and provides a limited object inheritance. The standard schemas which ship with most LDAP directory servers can be extended but you should try to stick with the supplied schemas if possible.

Each entry has a globally unique identifier called the Distinguished Name, or DN, which is a path along a tree structure containing nodes like country, organization, department, and name. The DN is public information, so it should not contain privileged information such as a person's SSN. A Relative DN (RDN) is a shorter part of the DN which is locally unique (such as within a company); an example is a userid or uid. When designing a Directory Information Tree (DIT), try to keep it as flat as is reasonable, because later changes will be easier to accommodate.

LDAP will make your life easier, but you will have to treat it like any other enterprise infrastructure element, and make it robust. LDAP can be used to integrate several information sources (HR, facilities, NIS, NT Domain, email, mailing lists), but you will need to get buy-in from all the interested parties.

One advantage to implementing LDAP is that you can assign ownership of different slices of the data to various groups, and delegate management of the data to those groups. This reduces load on the help desk. Different LDAP vendors implementations handle access control lists (ACLs) differently (as it is not part of the LDAP spec). The iPlanet LDAP server, for example, provides very powerful ACL mechanism and default behaviors (such as automatically giving the owner of a mailing list "ownership" to that entry).

LDAP can be used to replace or back end NIS for UNIX user information. Solaris 8 supports LDAP in the /etc/nsswitch file out of the box. There are scripts available to migrate your NIS databases to LDAP, and the schema is described in RFC 2307. On UNIX implementations which support Pluggable Authentication Modules (PAM), there are modules which allow direct LDAP binding over SSL (no passwords over the Net in the clear).

When you have disparate data sources, you may want to investigate one of the commercially available LDAP metadirectory solutions. These allow for bi-directional synchronization of data from multiple LDAP directories and other legacy sources (via "connectors") automatically, and are designed to resolve namespace conflicts. Depending on the complexity of your problem, you might be able to build your own metadirectory, but most of the time you will have to buy one. There are not currently any open source metadirectory systems. Most commercial metadirectories are extensible, allowing you to write your own connectors.

Another approach is to gateway legacy data sources into LDAP. A common example of this is ypldapd, which allows you to store your NIS maps in LDAP but uses traditional tools and clients to access it. This tactic should only be used as a transitional tool rather than a solution.

The change log is used for server replication and can be a nice hook into your directory server to accomplish some metadirectory functionality. The change log is data in the LDAP directory itself, and is protected by ACLs, but in an all-or-nothing fashion, giving a possible exposure of data that is otherwise protected with more granularity in the directory. Keep the change log protected. The change log can also be used for disaster recovery if you back it up.

Exporting LDAP data can accomplish a "poor-man's metadirectory." You can use scripts to massage LDAP-exported data into NIS maps, DNS zone files, or whatever. It's easy and it's fun!

NT presents special difficulties to LDAP deployment, especially because Active Directory wants to be in control. Microsoft also made some poor decisions regarding RDN and uses a proprietary password encryption scheme, forcing the use of plaintext passwords. Having a single namespace for UNIX and NT users will help you avoid various problems.

Mapping Corporate Intranets and Internets

Bill Cheswick, Lumeta Corporation

Summarized by Steve Hanson

Many of us often wonder what it would be like to leave our safe jobs in the corpo-

rate world to start a company. Part of Bill Cheswick's talk addressed this since he has recently spun off Lumeta from Lucent, starting a company to map corporate intranets as a service.

Most of the talk, however, was about the work Lumeta is doing. Some of this work developed out of infrastructure protection done for the government. Most intranets are completely out of control, and nobody really knows anymore what is on their networks. Some preliminary work is being done on mapping intranets and making simple visualizations of the systems. Lumeta's work was compared to some other projects in these areas, such as MIDS and CAIDA. Different visual representations of the intranets and the Internet have been done. Lumeta also did some work for the government during the Yugoslavian crisis, mapping the Internet in Yugoslavia, which gave good indications of what portions of the country had had their power knocked out by bombing.

One of the most interesting aspects of the talk was the maps themselves. Some of them are quite beautiful, and some have been purchased by museums as artworks. Of course, this information is being used primarily in the research community, and Lumeta hopes a large commercial market for their services will develop.

Maps: <*http://www.peacockmaps.com*>
Information: <*http://www.lumeta.com*>

The Design and Implementation of Highly Scalable Email Systems

Brad Knowles, Belgacom Skynet SA/NV

Summarized by Brian Baggett

Knowles, former email admin for AOL, gave a great talk on the problems faced in developing and running large-scale email systems. His talk focused less on implementation and more on fundamentals and architecture.

The approach taken by academia is often different from that of commercial institutions. Academia often reinvents the wheel and does things on a small scale, but occasionally it does something revolutionary. By contrast, the commercial world has no problem buying solutions; the time it takes to bring a product to market is crucial, and so the process tends to be more evolutionary than revolutionary. Most of their revolutions focused on scaling.

The underlying problems with all of the potential solutions are that none of them scale to handle 1 million-plus users on their own. Eliminating single points of failure or getting away from inefficient technologies like NFS has proven too difficult for many. Knowles summarized the pros and cons of POP3 and IMAP and identified the big bottleneck that hampers mail server scaling (I/O). This was a highly informative talk chock full of interesting facts and data, the results of which can be found at *http://www.usenix.org/events/lisa2000/invitedtalks/knowles.pdf*.

SANs — From A to Reality

W. Curtis Preston, Collective Technologies

Summarized by Steve Hanson

SAN systems are becoming a fact of life in most production environments, but it is not always clear if the SAN systems being purchased are serving a purpose or are just being bought because they are the new and hot technology.

Preston's presentation on SAN systems discussed the different technologies involved in SANs, and how to decide if SAN technology is a good fit for your needs.

He described the three competing distributed storage technologies – WAN, NAS, and SAN, and then went on to discuss the different hardware configurations available for SAN. The advantages of SAN (easy allocation, sharing, backup, etc.) were covered, as was its cost efficiency.

Finally, this talk discussed some of the current pitfalls of SAN (need to test thoroughly, incompatibility between different hardware), and how to decide if SAN is for you.

Why the Documentation Sucks, and What You Can Do About It

Steven Levine, SGI

Summarized by Josh Simon

Levine spoke and sang about documentation. Steven, a technical writer, talked about four major subjects: myths, difficulties, projects, and improvements.

First, Steven talked about some myths. One is that writers are editors. In reality, writers not only edit stuff others (such as developers) write, but write original content, maintain other existing documents, and produce both hardcopy and online help and Web-based documentation. They also coordinate and organize and are detectives; they have multiple information sources and work with various groups or departments. They are also responsible for documentation consistency and legal issues.

He then discussed some of the difficulties that writers face. He started this section with a song, and yes, all 300-plus audience members were singing along with the chorus. The major difficulties are lack of resources, conflicting perspectives, little (if any) usability testing of the documentation, shorter release cycles, distinctions in hardware and software, the problems of writing from experience on as-yet-nonexistent products, and having to rely on developers' time and interest to improve the documentation.

Third, he discussed some of the typical projects that writers are involved in. They are responsible for not only administrative documentation but also online documents, procedures and examples, and man pages (which may not be sexy but are certainly very useful).

Fourth, Steven discussed how to improve the documentation. The short answer is it's a two-way street. If you see something needing work in a document, let the author know. There's always some form

of contact information (even if it's postal mail). Document what you want solved. Document what you did to work around a problem to help others not have to go through it themselves. Formalize your informal people networks; if you're a developer, take your writer to lunch. Produce libraries of examples, procedures, tricks you use, and so on. Collaborate within the company across department lines. Collaborate with friends and acquaintances at other companies.

Refereed Papers

SESSION: DEEP THOUGHTS

Summarized by Socrates Pichardo

Theoretical System Administration

Mark Burgess, Oslo College

Mark attempts to demonstrate that system administration can be modeled in certain instances and that this theoretical model can be built and used to further understand system administration variables and their interdependencies. This is a very important concept; once you can identify all relevant parameters and model a particular problem you can proceed to optimize these variables to achieve maximum results.

Mark started his presentation by defining a simplified system administration model in which users, OSes, resources, policies, and states are the main components.

Mark's group has concluded that system administration problems fall into these categories:

- **Random behaviors, or type I models:** Variables follow random patterns within a well-defined cause-effect relation. Things like averages, median, and statistical theory can be used to optimize these models. Most stability problems will follow this behavior.
- **Anthropological behaviors, or type II models:** Variables follow anthropological behaviors within certain boundaries. Game theory and human conduct analysis can be used to optimize these models. Most utilization problems will follow this behavior.

In other words, influences on the systems can thus be classified as either random, stochastic, or passive (type I), or as intentional, adversarial, or strategic (type II), depending on the significance of the change.

An Expectant Chat About Script Maturity

Alva L. Couch, Tufts University

Couch presents his solution to current scripting tools and language limitations on solving complex system management tasks. To circumvent current limitations, he is developing an "interpreter" called Babble, which will take XML-like directives and mark up tags and interpret them into desired configuration tasks.

To accomplish this, he has created his own set of directives called Stream-Structure Markup Language or SSML, based on the "Jackson System Design" from the punch card era. Jackson's Principle claimed that the way to properly design a program for processing punched card stacks is to link the structure of the program with the structure of the stack that it processes. Alva expanded this principle into: "The structure of a fully functional interactive script is exactly parallel to the branching and looping structure of the device interactions in which it must engage."

SSML and Babble introduce a new level of instrumentation that will help system administrators tackle complex administration tasks but at the expense of extremely limited functionality and low reusability. Right now, there are only limited sets of complex tasks that Babble can address with success, and SSML scripts are very dependent on a particular revision version of the devices being managed. Each device revision requires a completely independent Babble script.

An Improved Approach for Generating Configuration Files from a Database

Jon Finke, Rensselaer Polytechnic Institute

Much of Rensselaer's site configuration information is stored in a relational database. In the past lots of little custom C programs and scripts were needed to extract this information in the appropriated format for server and daemons. This approach proved to be difficult to maintain and expand.

Maintenance was cumbersome due to the variety of scripts techniques and programming styles found at the site. Expansion into new operating systems was time-consuming since all these scripts needed to be "ported" to new hardware and operating system revisions.

Now, Rensselaer is using a centralized approach to this problem. They have gathered all logic and intelligence needed for configuration file generation into their Oracle database by using PL/SQL as their scripting tool. They generate all their configuration files in the database itself, then write a short program to dump the "files" out of the database and into the file system of the target machine. This allows all of the file generation code to be stored and maintained in the central database, using a consistent set of tools. In addition, the program to copy files could be generic, and once built for a particular operating environment, could be used for any of the files they might generate for that system.

Rensselaer's solution proves that centralization, when done right, can have a significant positive impact on the overall manageability of an IT infrastructure. Their usage of commercial tools (i.e., Oracle PL/SQL) makes their tools and scripts a bit more difficult to implement. At the end of his presentation, Jon made a "call for help" for anyone interested in "porting" their scripts into any of the open source products (i.e., MySQL).

SESSION: YOU, A ROCK, AND A HARD PLACE

Summarized by Craig Vershon

FOKSTRAUT and Samba — Dealing with Authentication and Performance Issues on a Large-Scale Samba Service

Robert Beck and Steve Holstead, University of Alberta

Robert and Steve noticed a performance problem with the Samba server that was being used as a gateway to AFS. The system was getting an unanticipated new load that couldn't handle all the users' authentication.

The Samba server was running as an AFS client gateway to things like Windows clients. They found they had to run Samba with clear text passwords enabled with password crack, but found an issue with some Windows clients sending passwords in all caps. This would allow them to authenticate to Samba but not to Kerberos, which requires more varied passwords.

Once they implemented the server, they found it to be highly CPU-bound. The server was receiving repeated password failures. They found a pattern: three bogus attempts, then the real password was sent. Windows was sending the "windows" password instead.

The solution: FOKSTRAUT, patches for Samba to make a DBM password cache. First, it caches the password that failed. It stores this in a database and keeps a failure count. After three failures, it checks again and resets to zero after success. Then they cached the "corrected case" success. This was stored in a clear text database. They found this "evil" and unsecure, but a compromise had to be made somewhere.

Available at:
<ftp://sunsite.ualberta.ca/pub/local/people/beck/fokstraut>

Improving Availability in VERITAS Environments

Karl Larson, Tellme Networks; Todd Stansell, Certainty Solutions

Karl and Todd spoke of problems and solutions they found in their VERITAS implementations.

Some of the tools they created or used:

- vxstat2gnuplot: this program converts the output of vxstat into a useable format for gnuplot. I thought this was really useful to see a graphic version of the disk/volume usage.
- cricket: used for trends-based monitoring
  <*http://cricket.sourceforge.net*>
- save-vxlayout: this script saves VM config details; good in use for disaster recovery info
- synch: from EMC; retrieves Symmetrics internal configuration details
- emcprints: shows all back-end controllers, devices, etc.; adds this info to the vxprint output.

They found problems with millions of small files on a single file system. The metadata becomes a performance bottleneck with VxFS. It uses metadata for intent logs (journaling). Changing millions of files causes changes to much more than just the files themselves. By default all the metadata is stored at the beginning of the file system. The space reserved could be too small. They found that by using Quicklog, they could move and store the metadata on a separate device.

Running backups on millions of sequential files daily makes it hard to obtain consistent "point in time" backups. They came up with two fixes: Volume Manager snapshots and using file system snapshots.

Cool tips and tricks for VERITAS: <*http://www.vxideas.org*>.

Designing a Data Center Instrumentation System

Robert Drzyzgula, Federal Reserve Board

Drzyzgula outlined the context of his problems:

- Several conflicting production cycles
- Tight deadlines
- Enormous economic models
- Highly variable capacity requirements
- Capacity is higher priority than reliability

Drzyzgula seems to have a limited budget with which to work. They purchase many parts in bulk and assemble systems themselves. He has been working on designing and building a monitoring and controlling device, to be used on all his various systems in the data center. The goals were to be able to monitor such things as power usage and temperatures and to be able to control power cycles, console access, etc. He spoke about the various chips, sensors, and other hardware he was using to attempt to implement this. He has been able to get a small working prototype to work in a limited environment.

SESSION: USERS AND PASSWORDS AND SCRIPTS, OH MY!

Summarized by Sam Shaffer

User-Centric Account Management with Heterogeneous Password Changing

Douglas Hughes, Auburn University

Hughes details development of a Web-based tool to allow students to change their UNIX and/or NT passwords. (NT authentication is via Samba.) The "User-Centric" in the title indicates that the system was developed with inexperienced students in mind.

The paper lists five similar systems and their principal parameters. Douglas indicated that a lot of information which might have made the other systems valuable was not available to him. He is looking for someone to maintain the code,

which is available at <*http://www.eng.auburn.edu/~doug/second.html*>.

Pelendur, Steward of the Sysadmin

Matt Curtin, Interhack Corp.; Sandy Farrar and Tami King, Ohio State University

This paper documents what seems to be a rather thoroughly automated account management system in use at Ohio State University.

Bottom line, this system modifies student and other accounts in response to changes in the university class database. Implementing it has greatly reduced the amount of time required to add and delete about 15,000 accounts per school term. The system is not available to the public because it isn't ready for prime time. As such, it is effectively "yet another set of design guidelines for an account automation tool" and provides some specifics of the implementation that might allow someone else to design and implement a similar system again.

Network Information Management and Distribution in a Heterogeneous and Decentralized Enterprise Environment

Alexander D. Kent and James R. Clifford, Los Alamos National Laboratory

The paper presents another unique set of circumstances which had to be incorporated into a user-friendly tool to allow people to manage their own data (such items as email aliases and email server passwords). There are several interesting components of this system, though. One is that changes to the database cause notification to an "event trigger daemon," which uses inetd to induce an update to an LDAP server.

The system source may be available. The paper indicates that U.S. encryption export issues require that the source be controlled. Make requests to the authors for information on requirements.

SESSION: THE TOOLSHED

Summarized by Jim Flanagan

xps: Dynamic Process Tree Watching Under X

Rocky Bernstein, Breakaway Solutions

In a talk as dynamic as the topic matter, Rocky described xps, which provides a view of the process table laid out as a tree rooted at the init, with colors distinguishing processes by owner or by state (running or waiting on I/O). What processes are shown can be determined by user-specified filters, and clicking on the process names can run a user-specified program such as ps to get specific information, or lsof to get the files open by that process.

The current version of xps is written using the Motif toolkit, but work is in progress to port it to GNOME. Much of the talk was spent weighing the various virtues and drawbacks of GNOME versus Motif, and how the GNOME view of the world changes how the xps problem is attacked and how one goes about optimizing for efficiency. GNOME also has tools which make life easier, such as Glade which builds dialog boxes that are much prettier than Motif's.

Rocky then demonstrated an instance of where xps might give more insight than traditional tools like ps or top. He went to a source directory with a fairly complex make procedure, and typed make. The make process showed up in the xps display, and alongside (or "under") that, you could see all the subsidiary awk commands and other commands being spawned to do the work and disappearing dynamically.

One questioner asked about the poll-based nature of xps having to read through the entire process table each refresh, and if it would be more efficient to somehow detect or trap calls to fork and vfork. Rocky said that it wasn't clear how to go about that, and that if you could, you might introduce problems with modifying the state of a program rather than simply monitoring it.

Extending UNIX System Logging with SHARP

Matthew Bing and Carl Erickson, Grand Valley State University

As grad student admins of 30 machines, the authors were being overwhelmed by having to try to winnow the interesting data from the volume of syslog data that was being generated. After giving a brief overview of how syslog works, Matt went on to present a list of areas where syslog could be improved.

- The routing of syslog messages depends only on their priority, which is the combination of facility plus a level, which is not very flexible. You can't add facilities to syslog.
- There is no standard message structure after the timestamp and pid.
- The priority information is not written to the logfile, and so is lost.
- In a centralized logging architecture, the timestamps are those of the originating system, and if the clocks on those systems are not in sync, event correlation is not feasible.
- It is not possible to detect if the logfile has been tampered with.
- UDP. Say no more.

Log-watching systems like swatch or logwatch don't completely compensate for these defects in that they don't provide realtime analysis or maintain state between runs, so they can't detect recurring problems and change their behavior (like stop paging you for the same problem).

SHARP (Syslog Heuristic Analysis & Response Program) is the author's response to these problems. Rather than replace syslogd, SHARP provides a daemon (sharpd) which also receives a copy of every syslog message. Modules compiled against the SHARP library can then register to get messages of various priorities. The messages which the modules receive are timestamped by sharpd for

event correlation. The modules can then do anything with the message: put it in a file, alert a user, or bounce the message back to syslog at a different priority.

Examples of modules were:

- Mark: expects a message from each machine at a certain interval, and logs a high priority message if it doesn't get it
- UserAlert: learns about users' patterns of logins (time and location) and notices behavior changes
- ProblemAlert: after a number of repeated messages of a certain priority, they will start getting sent back through syslog at a higher priority.

While SHARP will work with syslog, the authors recommended nsyslog as a replacement to work with SHARP, as nsyslog preserves the priority of messages, uses TCP and SSL to prevent spoofing, and uses chained hashing to prevent modifications of the logfile.

Planned developments include a Perl interface for modules, access to global configuration across all modules, and making SHARP completely thread safe.

Peep (the Network Auralizer): Monitoring Your Network with Sound

Michael Gilfix, Tufts University

[Winner of the Best Student Paper Award]

The Peep tool is an experiment in leveraging innate human skills in distinguishing subtle deviations from the normal state. Peep uses a continuous, non-intrusive audio representation of the health of your network in terms of events, state, and heartbeat types of sounds. The sounds selected for use with Peep are all from nature (waterfalls, bugs, birds, etc.) since this was thought to be the least intrusive.

Peep is a departure from the current state-of-the-art monitoring tools in that, where current tools are visual, problem-centered, and provide negative reinforcement, Peep is aural, normalcy-centered, and provides positive feedback. Peep attempts to remain ambient to take advantage of unconscious processing, and the sounds are all mixed together so that combinations of sounds become significant. If your network sounds like it sounded yesterday, than everything is fine.

Peep has a Producer/Consumer architecture which supports either distributed or centralized configuration. It is UDP-based, uses auto-discovery by both clients and servers, and employs leasing to handle servers that go offline.

Michael gave a demonstration of Peep. First, he played discrete sounds such as bird chirps which corresponded to incoming and outgoing mail, bad DNS lookups, and telnet connections. Then there were some continuous sounds like running water and general forest insect noise, which represented load average and concurrent users, respectively. Then he played a sample of actual Peep output for low load, which sounded like being in the forest near a stream. After that he played a high load average sample, and while I didn't feel like it was quite time to start filling sandbags, it seemed a little less comfortable.

Someone from the audience commented that for rare events, you might forget what sound went with what event. Michael said that they were looking for a solution to that, such as a GUI quick reference utility. Another questioner asked what the overhead was on the server side. Memory is really the biggest bottleneck, as all sounds are loaded into memory. Most of the processing overhead is in the sound mixing. In trials, they were only able to drive the server load average up to 0.6.

SESSION: 1984

Summarized by Socrates Pichardo

Thresh – A Data-Directed SNMP Threshold Poller

John Sellens, Certainty Solutions

Thresh is a simple SNMP monitor tool that lives in between realtime alert monitoring systems (i.e., Big Brother) and trend analysis and history tools (i.e., Cricket). The power of this tool lies in its simplicity. Thresh is an elementary but elegant implementation of SNMP monitoring services with an emphasis on easy configuration, low system overhead, decent notification, and some basic history and logging facilities.

In spite of Thresh's low system overhead, it has some scalability issues. Its main constraints are the lack of parallelism, configuration complexity, and notification throughput.

If you need some basic SNMP monitoring without all the hassle of configuring and maintaining a feature-rich SNMP console, then Thresh could be the tool for you; otherwise stick to some of the more capable SNMP consoles available today.

eEMU: A Practical Tool and Language for System Monitoring and Event Management

Jarra Voleynik, eEMUconcept Pty Ltd.

Voleynik described eEMU as a monitoring and management event console. eEMU is a client-server system that provides for rapid development of monitoring agents. Beside its console capabilities, eEMU has a scripting language that takes advantage of heuristic algorithms implemented at the server.

One of the things that sets this tool apart from the market leaders (i.e., TNG, Patrol, OpenView, and Tivoli) is the way it handles, aggregates, and presents alarms. While other console solutions will rely on color code representations and multiple windows of information, eEMU uses textual messages for each

event in a simple intuitive interface called eEMU browser. By default, the eEMU browser display only resources in "alarm state."

For its implementation, eEMUconcept Ltd. has decided to write its own eEMU agents and have them communicating with eEMU servers by using the eEMU protocol. eEMU works on the premise that all status information is handled by the eEMU server; therefore eEMU agents are simple scripts or programs that use the emsg program to send messages to the server.

One debatable design characteristic is their usage of TCP and not UDP for client-server communication. By using TCP and not transmitting "Systems OK" messages, they can avoid the common "UDP" storms generated by SNMP consoles and their polling efforts. On the other hand, we can argue that the overhead generated by TCP connections as well as the lack of "Systems OK" messages could produce some challenging programming problems for the developers and minimize their utilization gains. On their labs, they have been able to monitor 100 systems on a 33Kbps dialup line or 1,000 messages a minute on a 400MHz Pentium PC.

The power of the eEMU messaging language can be easily illustrated on the eEMU agents. eEMU agents with a few lines of code can handle complex monitoring scenarios.

Finally, eEMU has been successfully integrated with some of the major monitoring software vendors. This integration can be accomplished by using eEMU action scripts as well as other scripting hooks to their event engine.

Aberrant Behavior Detection in Time Series for Network Service Monitoring

Jake D. Brutlag, Microsoft WebTV

Realtime monitoring of service networks can generate vast amounts of time series data. Open-source tools like RRDtool and Cricket can help you with collecting, storing, and visualizing this data but, for large networks, you still need a methodology or tool to help you identify failures and/or abnormal situations.

Microsoft's WebTV division was facing this problem, and the amount of data being generated was enough to distract their network administrators from the important issues facing their networks. Their solution was to integrate a model based on exponential smoothing and Holt-Winters forecasting into the Cricket/RRDtool architecture.

Their model takes into consideration the following characteristics of time series data:

- A trend over time
- A seasonal trend of cycle
- Seasonal variability
- Gradual evolution of regularities over time

The Aberrant Behavior Detection model unpacks into three pieces, each building on its predecessor:

- An algorithm for predicting the values of a time series one time step into the future
- A measure of deviation between the predicted values and the observed values
- A mechanism to decide if and when an observed value or sequence of observed values is "too deviant" from the predicted value(s)

This model was implemented by enhancing the RRDtool with five new "consolidation functions":

- **HWPREDICT:** an array of forecast computed by the Holt-Winters algorithm, one for each Primary Data Point (PDP)
- **SEASONAL:** an array of seasonal coefficients with length equal to the seasonal period
- **DEVPREDICT:** an array of deviation predictions
- **DEVSEASONAL:** an array of seasonal deviations
- **FAILURES:** an array of Boolean indicators

On the Cricket side, Cricket 1.1 already includes a new type of monitor-threshold specific for aberrant behavior detection. Combining these two tools, they were able to monitor and alert on aberrant behavior conditions.

SESSION: THE SORCERER'S APPRENTICE

Summarized by Vinod Kutty

PIKT: Problem Informant/Killer Tool

Robert Osterlund, University of Chicago

PIKT is a system configuration management tool, addressing problems that tools such as cfengine are designed for, but in a more general purpose way. It monitors and warns of system problems, and has features that allow it to take corrective action if needed. The focus is on managing large numbers of machines rather than on individual machines.

Sysadmins typically write custom scripts to address these issues, but problems of OS diversity, code robustness and maintainability, specificity to certain tasks, scheduling scripts, error logging, script and configuration file distribution, and so on, plague this approach.

PIKT is designed to solve a lot of these problems in a fairly platform-independent (i.e., UNIX-flavored) manner. At its core is an embedded scripting language and a configuration file pre-processor that can be used with languages other than the PIKT language. It also includes a scheduling system, a distribution mechanism (like rsync/rdist), and a remote process execution facility (like rsh/ssh).

The typical deployment involves a central "master" machine which controls "slaves" (i.e., clients). Configuration files are managed on the master, then run through a tool that pre-processes and installs files, pushes changes to slaves,

executes remote commands, and so on. A scheduling daemon on each client runs alarm scripts to monitor various aspects of a system (e.g., disk usage, running processes, etc.), and a flexible logging system is provided. There is some client/server security implemented using secret-key host authentication.

Some use cases not directly related to monitoring include installing and managing non-PIKT scripts and configuration files (e.g., `inetd.conf`), document distribution, and managing security (by complementing security tools with logfile analysis, security configuration file maintenance, and so on).

Future work includes a full security audit, a standard library of configuration files, a rewrite of the PIKT script interpreter (possibly using embedded Perl or another language), improved message routing, and graphical interfaces for the piktc and alert management components.

Relieving the Burden of System Administration Through Support Automation

Allan Miller and Alex Donnini, Hands-Free Networks

Companies increasingly have to support a growing population of users with minimal application or other technical training. This in turn increases the burden on support organizations.

An automated support system can help avoid a crisis and improve the scalability of the support organization. However, automating support is difficult and often leads to a "mountain of kludges" that do not exhibit an understanding of the issues. Automation is best suited for repetitive tasks and touches upon all aspects of a system.

Traditional user support involves some kind of problem/ticket system with a database back end that stores solutions to previously encountered problems. At each step of the support process, human intervention is necessary to clarify end user symptoms, search existing symptom + resolution databases, escalate the request, and so on. This is a labor-intensive, error-prone process and often relies on mental knowledge rather than a database.

The automated support system under discussion uses a software client, instead of the user, to detect and report problems. A database is used to track symptoms and resolutions that include executable code (called "scrips," which are collections of various modules). Thus, the software client can automatically resolve the problem if there is a match.

The expectation is that about 80% of all problems can be solved this way, with the remaining 20% involving an escalation procedure. In addition, there is a well known 80-20 rule in support circles that suggests the size of the database that can solve 80% of all problems is expected to be about 20% of the size of the universe of solutions.

Experience with the system so far has been on Windows operating systems, with a Linux port in the works. Some beta sites are using the software, and feedback indicates a remarkable similarity in problems encountered and automatically solved, despite considerable differences in the businesses.

Additional uses envisioned for the system include automated administration and maintenance functions, security patch distribution, and automated support for mobile and embedded systems.

FTP Mirror Tracker: First Steps Towards URN

Alexei Novikov and Martin Hamilton, ITEP

The FTP Mirror Tracker package attempts to decrease the load of FTP traffic on WANs while improving performance for end users. It does this by localizing FTP accessible files on mirrors and employing a transparent scheme to redirect users to the nearest mirror with the latest, most complete copy of all the files needed.

A robot gathers directory listings from FTP servers, and a summarizer component parses these and creates MD5 digests on a per-directory basis. A database back end using MySQL keeps track of FTP Mirror Tracker data and links collections to the domains being tracked. A digest exchange compresses and moves the digests into a Web-accessible area (for other trackers to access), and front-end programs provide the means for users to query trackers. An ICP (Internet Cache Protocol) server was also written as a means to allow cache querying by other Web caching systems.

This comes into play when users are redirected to the closest FTP mirror, by using Squid to redirect URLs to the ICP server component for rewriting.

Internal support for URNs (Uniform Resource Names, i.e., a persistent, location-independent naming scheme that decouples location from the name of the resource) has been added to FTP Mirror Tracker.

The system has been put into production, and preliminary results show a reasonable cache hit rate, but improvements are expected. Some of the functionality implemented on top of Squid have been folded into Squid itself as of the distribution of version 2.3 .

SESSION: FULLY AUTOMATIC

Summarized by Socrates Pichardo

Deployme: Tellme's Package Management and Deployment System

Kyle Oppenheim and Patrick McCormick, Tellme Networks

[Winner of the Best Paper Award]

Deployme is Tellme Network's solution to manage the package update life cycle across a large number of independently configured hosts. It is highly flexible and

has been extended to handle many different types of packages. These packages include standard UNIX tools, local applications, Web site content, and voice site content. Deployme value can be maximized on packages that require fast, frequent deployment.

Deployme's mission is to provide a central system for tracking the entire life cycle of software packages. Its designing goals are:

- Support a wide audience
- Robustness
- Augment the development process
- Flexible destinations
- Efficient use of network bandwidth
- Quick pushes
- Seamless activation
- Rollback
- Scalability

On the other hand, Deployme designers intentionally left out several features while pursuing simplicity and shorter time to market. These features are:

- No local package management
- No dependencies
- No fine-grained operations control

Deployme is written entirely in Perl 5 and has a simple three-tier architecture.

Although Deployme is a well-implemented solution to Tellme Network's package management problems, it lacks certain features/services in certain areas. The authors mentioned some of Deployme's shortcomings:

- It doesn't have the concept of "sites" or several machines sharing a physical location.
- The lack of "transactions" at the database level makes it difficult to accurately determine the integrity of the data after a system failure.
- Deployme's lack of multicasting support negatively impacts network utilization when the same package is sent to a large number of servers.
- The system has no security features as of version 1. There is no control over who, what, or where.

Tellme Network is currently working on solutions for many of these shortcomings.

Automating Request-Based Software Distribution

Christopher Hemmerick, Indiana University

Netdist is a very complete solution for software distribution that was designed from the ground up with security, modularity, flexibility, and extensibility in mind. Netdist provides an automated mechanism for system administrators to request and receive software exports with an immediate turnaround. The system provides a simple user interface, secure authentication, and both user- and machine-based authentication. Each of these is configurable on a package-by-package basis for flexibility.

Netdist is a modular service. The user interface, authentication, and authorization are independent of the export protocol. The author is currently distributing via NFS, but adding an additional protocol is as simple as writing a script to perform the export and plugging it into Netdist.

Netdist is implemented using Perl 5 and some modules from CPAN, PGP, cron (or any other job scheduling service), and an instance of Apache with at least mod_perl and preferably a module for secure transactions. The NFS export control scripts have been written for Solaris but could be easily ported to other UNIX flavors.

The only shortcomings of this tool are its lack of installation scripts and availability. Netdist is still pre-alpha, and a lot more work is needed in order to ease installation. Also, several of the Perl modules and scripts do have host- or port-specific information coded into them. Although each of these instances is documented, the authors will attempt to extract all these values into a configuration script in the next version.

Use of Cfengine for Automated, Multi-Platform Software and Patch Distribution

David Ressman and John Valdes, University of Chicago

The author's main requirements were to create or buy an automatization package for software and patches distribution in order to improve the level of services being provided to their end users and to liberate their two SAs (authors) from these repetitive tasks. Some of the important characteristics of the solution were cost, ease of use, current development, and security.

Their solution was to "glue" with Perl some of the "best of breed" tools available for the different tasks. They took Cfengine (Configuration and System Management tool), NFS (for their file system exports), and RPM (Red Hat Package Manager) and used them as building blocks, in addition to Perl and mySQL, to create the Web interface as well as the back-end database.

The outcome for the software distribution problem is a Web-based front end where users can request which software package they want to install. Once they submit their request, the system will insert these requests into the database, create the necessary exports, and offer users the opportunity to launch the RPM module requested. On the patch distribution side, hosts will check periodically with the software depot server for new patches available for their OSes and architectures. Once clients find new patches, they will proceed to install them and to report back the exit code of such installs.

Future work on this project will involve expanding its services to include OS upgrades as well as support for other UNIX flavors.

SESSION: BUILDING BLOCKS

Summarized by Vinod Kutty

Unleashing the Power of JumpStart: A New Technique for Disaster Recovery, Cloning, or Snapshotting a Solaris System

Lee Amatangelo, Collective Technologies

A production data center requires processes to reduce downtime of critical servers as far as possible. Apart from hardware/software high-availability solutions, a disaster recovery plan is a must.

This paper describes a system that provides the following for Solaris systems:

- Bare-metal recovery
- Creation of a system snapshot on optical media
- Cloning of a system
- Rollout of multiple system clones

A typical application is one in which the root drive(s) of a server has failed or been corrupted by human error or hardware failure. Traditional backup/restore methods are not feasible when the OS cannot run.

One approach is to use bit-level imaging (or "ghosting" in the PC world), which can be quite fast, but not as flexible as something like JumpStart, which is the alternate approach that performs automated Solaris installations.

The solution – the Capture and Recovery Tool (CART) – combines both techniques. The tool evolved in an environment where security was important, and this is reflected in the requirements:

- No magnetic media allowed
- Recovery media must be bootable
- There should be minimal user interaction
- Multiple sets of removable media must be handled (e.g., if a snapshot's size requires three CDs)
- Should not involve directory services such as NIS, NIS+
- Should not depend on NFS

The implementation depends on the Solaris installboot command – which can place boot blocks on optical media – and the customizable nature of JumpStart.

A good understanding of JumpStart operation is required to understand the customizations made, but the important points are:

- Although typically associated with network installs, JumpStart is also a part of traditional installs of Solaris from CD-ROM, installing from local media rather than the network
- CART plugs into this JumpStart mechanism and replaces certain scripts so that JumpStart does not perform normal installation of packages, patches, and so on. Instead, it provides enough of a boot up process to get to the stage where a CART script can be run, after which JumpStart relinquishes control to CART

Future enhancements to CART include integration into a networked environment and implementation on other UNIX variants.

A Linux Appliance Construction Set

Michael W. Shaffer, Agilent Labs RCS

The motivation for this project started with the author's need to support remote installations of Linux servers providing file, print, and network routing services located in areas with few skilled personnel capable of disaster recovery.

One way to address the support issue and establish a fairly error-free disaster recovery process is to eliminate the traditional install of the operating system and additional software, and instead boot and run directly from removable media, using a Linux distribution configured for this purpose.

Rather than independently tuning a Linux distribution for each specific purpose – such as a print server – the scope of the project was enlarged to create a more generic framework for creating minimal Linux systems. Hence the name "Linux Appliance Construction Kit (LxA)", where the 'x' represents the function of the appliance (e.g., LPA == Linux Printing Appliance).

The design and implementation of LxA followed several principles:

- Systems are built by composition of needed pieces rather than reduction of a large set of components
- Systems run from read-only and/or removable media (although hard drives may be used for swap, /var, /tmp and other transient storage)
- Omit login and run-time configuration (except during development, where facilities such as console login and an interactive shell may be needed for debugging)
- Use modern, standard components, such as the kernel, libc, and so on

A lot of work goes into determining what is needed, testing the images, creating bootable CDs/floppies, and so on. The underlying technique for running an LxA system is to use an initrd image and run the entire system from it at boot time.

There are numerous advantages to using LxA over the more general purpose Linux installations, and these were also important goals in the design:

- Reduced complexity, which in turn enables better documentation and a more thorough understanding of the system
- Reduced security vulnerability, resulting from simplicity
- Reduced setup, maintenance, and upgrade time
- Reduced probability and impact of hardware failures

Future work will address more types of LxA appliances, enhancements to the existing LPA-CD appliance, and automated scripts for identifying components needed for new LxA systems.

More information about LxA can be found at <*http://www.equusasinus.com/lxa/*>.

AUTOMATING DUAL BOOT (LINUX AND NT) INSTALLATIONS

Rajeev Agrawala, Shaun Erickson, and Robert Fulmer, Lucent/Bell Labs Research

Although tools are available to automate the installation of Linux and NT, there are no good tools to automate the installation of PCs that dual boot either Linux or NT, from separate partitions on a hard drive.

This motivated support personnel at Lucent/Bell Labs Research to design a solution to this problem, as users were already starting to use dual boot installations of NT and Linux. These were inconsistently installed by different admins, time-consuming to perform, and not reproducible.

The solution employs automated installs starting with a modified "bootnet" Red Hat Kickstart floppy. The alternative is to use disk cloning, but this requires similar hardware and peripherals, cannot deal with unique NT SIDs, and involves a lot of work in updating an entire image when any piece of software is changed.

The process is designed to start with an admin booting from a floppy and selecting an install option (NT 4, Linux, or both). For dual boot installations, the first OS installed is Linux, using Red Hat's Kickstart. Automated customization is performed, the disk is repartitioned, and a DOS file system is created for the second OS install, namely NT. Note that this file system is eventually converted to NTFS.

Some files required for the automated NT installation are copied to this partition, and a reboot occurs to invoke the NT installer. After the NT install and customization steps are complete, a reboot surrenders control to the first OS – Linux – where final configuration of X and audio must be done manually, due to problems with lack of device driver support that could interrupt the automated installation.

Some difficulties were encountered with passing information about the installation type to the Linux kernel, creating two primary partitions at once, and locating LILO in /boot vs. the Master Boot Record. The design accommodates solutions and/or workarounds for these where necessary.

The system has been in use for more than six months, and future plans include support for automated X and audio configuration in Linux, and the addition of other operating systems (e.g., Win 2000).

Source code and configuration profiles are available from the authors <*dualbootinfo@research.bell-labs.com*>.

NETWORK TRACK

DEPLOYING QUALITY OF SERVICE FEATURES ON YOUR NETWORK

Eliot Lear, Cisco Systems

Summarized by Paul Federighi

Eliot Lear's talk described what quality of service (QoS) is, why you might need it, and the methods for achieving it. The talk was mainly focused on QoS as it pertains to voice communication on an IP network, though other types of data such as video and Web traffic were also mentioned.

As Lear explained, QoS is a method of giving preferential treatment to certain types of data on the network. Applications such as interactive voice and video have special needs. Voice has certain bandwidth and latency requirements and is drop sensitive. Most packets must make it through with less than 200 ms of latency. This includes transmit time and any queuing delays. QoS is not important for non-interactive traffic or non-time-critical traffic that can be buffered.

Lear stressed the point that QoS features are needed on every piece of equipment in the communications path where packets can be queued. This includes routers, Ethernet switches, ATM switches, frame relay, etc.

There are two models for achieving QoS: integrated services (Intserv) and differentiated services (Diffserv). With Intserv, the application specifically requests (via RSVP) resources at every hop along the way. Call setup happens first, then receivers request reservation (in both directions). Some of the advantages:

- Works with both unicast and multicast traffic
- End points know early on whether there's a reservation

By contrast, Diffserv has no end-to-end signaling. Instead, it uses per packet marking rather than marking the entire stream. Traffic is marked based on a policy domain and is policed at the edges. Since there is no signaling, an error could cause an application to fail silently. One needs to pay careful attention to traffic engineering and either allow for additional bandwidth on links or constrain traffic to predictable paths. Packets are separated into different classes:

- Best effort
- Assured forwarding (AF) – all data will get there
- Expedited forwarding (EF) – preferred over other traffic

Lear explained several different buffering and queuing techniques on network equipment. The old way is to use a FIFO. When congestion occurs, the end of the transmission gets dropped. However this is unfair to lower bandwidth protocols, and dropping packets just wastes bandwidth. Methods for overcoming this include random early detection (RED), weighted red, priority queuing, and weighted fair queuing. Other methods were mentioned as well.

Management and monitoring are important with QoS. You need to know if your packets are getting through and if the latency is tolerable. Tools are just starting to become available.

Trying to achieve QoS across the Web has scalability problems. There is research

beginning with "bandwidth brokers." Right now you can't get QoS across multiple providers. Instead, a good idea is to distribute data throughout the Web to reduce latency and get better bandwidth utilization.

When asked about security, Lear responded with the point that you can't do packet classification on encrypted data.

When asked about features to look for in hardware, Lear mentioned several questions to ask, including, "Are there multiple output queues?" and "Can you classify data in the output queues?" It's also important to remember that bottlenecks typically occur because of the line cards used, not the backplane of the device.

SESSION: ANALYZE THIS!

Summarized by Tony Katz

Wide Area Network Packet Capture and Analysis

Jon T. Meek, American Home Products Corp.

Jon Meek's talk covered why we need to analyze and how he went about doing it. We need to know what is happening on the wire to diagnose such problems as slow applications and network congestion. To monitor the frame networks, Jon used a small PC running Redhat Linux, which was plugged into the CSU/DSU via a serial connection to capture HDLC packets for analysis over time. The topic of time interval was interesting. You can capture packets continuously, which takes a lot of resources, or in intervals, such as 15 minutes or 10 seconds. What interval you choose can make a significant difference. Jon did all of this using C and Perl. It is a fairly inexpensive way to monitor your frame relay and get reasonably good results.

Sequencing of Configuration Operations for IP Networks

P. Krishnan, IPSoft, Inc.; T. Naik, Bell Labs; G. Ramu, CoSine Comm., Inc.; and R. Sequeira, IPSoft, Inc.

P. Krishnan addresses the problem of losing segments when updating routing configurations across a complex network if the updates do not happen fast enough. The proposed solution is sequencing. This solution will work but it assumes many things, e.g., that you are using OSPF, that routes are static routes, etc. This is achieved by indirect telnet, traceroute, and reverse traceroute. Not bad if your environment fits all the criteria.

ND: A Comprehensive Network Administration and Analysis Tool

Ellen Mitchell, Eric Nelson, and David Hess, Texas A&M University

There are lots of vendors and even more software out in the world and each one does something important, but none of them do it all. ND was designed by Texas A&M to accomplish this task. They wanted it to be powerful but low level, portable, customizable, and scalable. It was written primarily in Python, and uses SNMP, SQL tables, and a MySQL back-end database. Python was used for its modularity. With ND you can enable ports, configure new devices, do scripting, but best of all is that it has built-in documentation. This is a great feature, not so much to point a finger at someone, but to know who to talk to about why a particular change was made. Texas A&M is also looking to add event monitoring to ND. Currently, ND is not available but will probably be released to the public sometime in the future.

SESSION: GO WITH THE NETFLOW

Summarized by Tony Katz

Combining Cisco NetFlow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics

Bill Nickless, John-Paul Navarro, and Linda Winkler, Argonne National Laboratory

The first part of the presentation given by Bill Nickless focused on the problem of having a high performance network with a minimal firewall.

Cisco's NetFlow provides a summary of data traffic through a router. This data must be captured and analyzed. Argonne's way to do this was through the use of database technology. Their hurdles are the amount of data coming in and the ability of the database to keep up. They went with a high-powered database running on an SGI Origin 2000. They experimented with both MySQL and Oracle 8I but ended up using a SQL back-end database. They used Perl scripts to catch the data and feed it into the database for analysis. This worked very well overall. The big issue is that not every site has an Origin to process thousands of records at a time. Scalability is determined by your database application and tuning parameters.

The OSU Flow-Tools Package and CISCO NetFlow Logs

Mark Fullmer, OARnet, and Steve Romig, Ohio State University

Steve Romig spoke about his application of NetFlow. Their interest was more of a security focus. They also had a voluminous influx of data but handled it a little differently. OSU looked at aggregation, collection, viewing, and security using a set of tools they created called Flow Tools. They reduced the data load by aggregating the data into summaries. This allows viewing at any given point. Their security features were the most interesting. There has been a lot of focus put on incident response. OSU used a

variety of Flow Tools to detect "interesting network traffic," host or IP range profiling, as well as detecting network attacks, i.e., denial of service. OSU is continuing to expand their set of tools to do more in the realm of the client/server role on the end points of the wire. For a closer look at the tool set, check their Web site <*http://www.net.ohio-state.edu/software*>.

FlowScan: A Network Traffic Flow Reporting and Visualization Tool

Dave Plonka, University of Wisconsin-Madison

Dave Plonka's presentation was saving the best for last. Not to say that the other implementations were bad, but they did not have a visualization component and Dave's did. The University of Wisconsin-Madison used an open systems software package called FlowScan. FlowScan uses a report module called CampusIO to accumulate the raw flows and push the statistics to a round-robin database. The data was then taken by FlowScan and graphed. The graph, which was color coded to traffic, was able to show both inbound and outbound traffic. The colors of the graph differentiated between HTTP, FTP, and (the ever-popular) Napster traffic. The benefits of the graph over data figures is that you can instantly see what type of traffic is using the most bandwidth at any point in time. Graphing also helps you pinpoint anomalies easily. Future expansion points for FlowScan are in the area of event notification or alerts. For more information go to <*http://net.doit.wisc.edu/~plonka/FlowScan*>.

This was a great depiction of the uses of Cisco's NetFlow. It is a definite improvement over analysis by sniffer.

Broadband Changes Everything

Summarized by Josh Simon

Brent Chapman, Great Circle Associates

Brent Chapman spoke about how broadband – which includes the variants of DSL, cable modems, and possibly even wireless – changes the way people perceive the Internet.

Broadband has two features: it's high speed and always on. DSL provides speeds on the order of 144Kbps (or more than 7Mbps). Cable modems share the same big pipe but provide similar high speeds. In comparison, even the fastest phone-modems provide no more than 53Kbps. By "always on," Brent means that there's no longer any dial-up delay and no busy signals. This makes the Internet like electricity or water: you flip a switch or turn a knob and it's just *there.* This will change how people perceive and use the Internet in the long run; rather than saying, "I'll go online later and do that," they're much more likely to hop on and off the Net for brief visits to accomplish tasks as they come up as opposed to waiting until later. (Note that most consumer electronics today – stereos, televisions, and microwaves – don't actually power themselves completely off. They remain in a reduced-power "stand-by" mode so they can appear to power up more quickly when needed.)

Broadband is also cheaper than traditional leased lines. A T1 line from a telecommunications provider (telco) used to run $1,500 a month. Comparable speeds via DSL are on the order of $300 a month.

The revolution in providing broadband leads to new capabilities, such as connecting small offices or home offices to the Internet at high speeds, as well as making telecommuting more effective for virtually everyone. It also leads to new services or more efficient older services, such as:

- Streaming audio and video
- Backing up over the network (such as @Backup)
- Software auto-updates (Apple, Symantec)
- Push services (PointCast)
- Cooperative computing (SETI@home)
- Interactive games

Unfortunately, broadband also leads to new security threats. "Always on" means "always vulnerable." You can no longer assume that you can only be hit by attacks when you're online in front of the computer when the Internet link is always up. Cable modem lines are shared within a neighborhood, so "Network Neighborhood" takes on a whole new meaning. If you have shared your disk or printer within your own home, you're also sharing them with the entire cable neighborhood. We should expect to see new hardware and software firewalls built into broadband DSL in the near future.

Broadband also allows you to save money. Many homes have more than two computers, so networking them within the home to share a single big pipe for bandwidth makes more sense to more users now. This means that you could cancel your second phone line (saving about $15/month) as well as multiple ISP accounts (saving $20/month).

What's coming in the future of broadband? Brent expects that virtual ISPs (for sales and marketing features), affinity ISPs (like credit cards), subsidization, and cross-marketing will happen in the near term. We'll also see voice-over DSL and voice-over cable (some areas already have one or both of these); the problem faced by the providers here is "five 9s reliability," or less than five minutes of downtime – scheduled or unscheduled – per calendar year. We'll see more network appliances (like WebTV and Tivo) and radio- and broadband-ready MP3 receivers. We'll also see Internet-enabled appliances, such as the refrigerator with a touch screen for restocking linked to a grocery delivery service such as Peapod or WebVan.

There are several IT management issues with broadband. First among these is security: should employees' homes be inside or outside the corporate firewall? If they're inside, who other than the employee has access to the company network? If they're outside, should the

corporate Internet access be shared with the homes? If so, we need to have some kind of firewall protection (but then who maintains and monitors those firewalls?); if not, the cost to the company will skyrocket since every home user needs to have their own bandwidth. What carriers are available to the employee? Who supports and supplies the home system? How can you provide mutually secure access, such as when an employee's spouse works for the competition? Is a VPN the right solution? If so, is it PC-based (which leads to driver issues) or router-based (which doesn't address the other-people issue)? Are personal firewalls the answer? Those also lead to issues of who provides, configures, reconfigures, manages, and updates them, and ignores the multiple-connection issue.

In the question and answer section, Brent noted that distributed denial of service attacks (DDoS) will increase. Host-based security has to come back into style, since firewalls are no longer enough protection. The Cheswick/Bellovin model of a crunchy exterior and creamy interior no longer applies. Satellite broadband is unlikely because of the huge latency involved. Broadband affects the core routers. When asked what it'll take to administer the high-bandwidth providers (such as Akamai), Brent noted that there's no good answer yet but we certainly need to work on it. As an example, Akamai has 600 servers and is moving toward 600,000 servers. Broadband also leads to more peer-to-peer networking, so the traditional source-and-sink model may need to be redefined.

Security Track

Cops Are from Mars, Gurus Are from Pluto: Dealing with "the Feds" and Other Cops

Tom Perrine, Pacific Institute of Computer Security

Summarized by Dave Homoney

Tom made this a very interesting talk. His injection of real-world scenarios was very helpful. The talk was geared toward sysadmins, those of us who might have a run-in with a hacker and need to know where to turn and the protocols to use. He also talked about what to do when law enforcement (LE) comes to you.

Tom described how to tell if a call from the FBI is a hoax: if an FBI agent is calling you directly, it probably is. He stated that most federal agencies will contact you through a local law enforcement agency. He also said that contact by the FBI would be through the nearest local office.

Tom mentioned several cases in which LE screwed up, particularly the case against Steve Jackson. This case produced a lot of negative press for LE and marked the point at which LE moved from the guns blazing approach to the computer savvy LE officer approach.

Tom also talked about when not to help LE, stating that the first thing you need is a good lawyer. You don't want to do anything as "directed" by LE or you could be considered an agent of the government, causing you problems in court. Instead, you should follow the directions of your company's legal staff. And, of course, you will need to comply with all court orders such as subpoenas.

Finally, you should create a bridge between yourself and LE so that when you need them, you'll have a friendly contact. He mentioned several times that "they are just like us," adding that there are always exceptions.

Does It Take the Same Skill Set to Secure a System as to Break into It?

Panelists: Peter Shipley, Lab OneSecure Inc.; Mark Hardy, Guardent, Inc.; and Elias Levy, securityfocus.

Summarized by Dave McFerren

Some of the topics the panel discussed were:

- Should companies hire crackers to catch other crackers?
- Can you trust someone who was once a cracker?

Discussion was fairly one-sided concerning the skill set needed to break in compared to the skill set to secure. The overwhelming majority of opinion was that cracking requires only a subset of the skills needed to secure systems; there are many different ways to compromise a computer, and a cracker needs only concentrate on one particular service that the computer may deliver. Another topic tossed about was the question of whether you can hire "black hats" to do "white hat" jobs. Although there are many startup or fringe companies that tend to do this, the general consensus was that you should become a white hat by yourself and make a foray into the corporate world before earning the trust of the "suits."

The most interesting discussion arose in response to the question, "What does it take to become a security expert?" One panelist – who had previously been a black hat – insisted that you had to breathe, live, and eat security for years to become good at it. Others disagreed, believing that you can have a life with family and friends and still be able to do a good job at security. But most agreed that since security requires more than the traditional 40 hrs/wk, a person would have to be at least somewhat obsessed with the subject to be really successful.

Overall, the discussion was interesting, although I was not sure I got more than the single perspective held by the computer security profession. But it did show

me the "other side" of the security issues that I deal with on a day-to-day basis.

Real-World Intrusion Detection – First Steps

Mark K. Mellis, SystemExperts Corporation

Summarized by Steve Wormley

Mark Mellis covered much of what is needed to set up intrusion detection using primarily free products on a small- to medium-sized network. He noted this discussion didn't apply to larger sites, because they generally have larger problems, but the basics are the same.

He first gave an overview of why one would do intrusion detection (ID). Basically the crackers are becoming more sophisticated, the networks are becoming more complex, more protocols are flowing through the firewall, and there are more connections to business partners and other points of attack.

The assumptions for this talk were that the solution needed to be cheap, that the SA was familiar with ftp and make and normal freeware/open source setup, and that the admin was busy and ID was a part-time job.

ID systems should tell you when real threats occur but be able to log even door rattling. Important things to trigger on are config changes, auth failures, attempts to probe the site, and attempts to access services.

The things he recommended deploying included centralized syslog functions (UDP syslog or nsyslogd); a log-analysis product like log_analysis or log surfer; a tripwire like Tripwire or aide; Klaxonto monitor for connection attempts on unused ports; sscanlogd to monitor for portscans; and a product like snort, which is a lightweight network IDS.

A couple of points to remember: routers are hosts too and need to be monitored and can use syslog to do so. To capture authentication failures, brute force does work. Don't forget application exploits; scan the Web and appserver logs for anything out of place.

Finally, all exposed and all infrastructure machines should be running host-based ID, and network ID systems should be put where traffic is both concentrated and sensitive. And, of course, anything that can be done is better than nothing.

This was a good overview of the products available and things that can be used in ID. It was a good talk for anyone who needs to install this type of service on a small scale or as a precursor to learning how to do it in a large environment.

Mark K. Mellis' URL is <*http://www.systemexperts.com*>.

SESSION: SOMEONE'S KNOCKING AT THE DOOR . . .

Summarized by Eric Lakin

Tracing Anonymous Packets to Their Approximate Source

Hal Burch, Carnegie Mellon University; Bill Cheswick, Lumeta Corporation

This paper was one of two papers to receive the "Best Paper" award for LISA 2000. It was based on work done a couple of years ago within Lucent's corporate network, concerned with ways to find the source of a denial of service (DoS) attack when the source of the packets is forged (the norm). Some of the assumptions made in the research which limit the usefulness of the technique against distributed denial of service (DDoS) attacks seen recently include the following: the source of the DoS packets is a single source, no modifications to the current network infrastructure can be made (router or protocol), the attack is long term, and the packet-rate is constant. Further, only DoS attacks that seek to overwhelm the victim with bogus packets are considered; attacks that attempt to cause a malfunction by specially crafted input are not.

When considering the network topology for the purposes of a DoS attack, it was convenient for the authors to describe it in terms of a tree graph. The victim's machine is at the root of the tree, with network nodes being nodes in the tree. Each path out of the victims local network subnet is a branch in the tree, with further branchings being paths out of the connected subnets, and so on. One path to a leaf node is the attacking host.

Because the source of the DoS packets is almost always spoofed, the assistance of the ISPs and network administrators outside the victim's network are usually required to locate and shut down the attacker. There is often little motivation for these people to help, and even finding the appropriate person to help may be a challenge. If the appropriate people are found and are willing to help, they can either put their routers into debug mode to determine which path the attack is taking or selectively cut off paths briefly and see if the attack slows or stops.

When the outside network managers cannot be contacted for some reason, is it possible to determine the approximate location of the attacker, without physical access to the outside network or their routers? The authors of the paper were able to come up with a way to selectively "deactivate" a line remotely by using the "chargen" service and UDP broadcast packets to selectively overwhelm, or DoS, a branch in the tree. By selectively overloading individual branches of the tree and watching the rate of incoming packets, one can determine with increasing accuracy where the attacker is located.

Because of the method used to overwhelm the branches, accuracy is limited to determining the subnet of the attacker. This, however, is enough to allow contacting of the appropriate ISP. In simulated DoS attacks within Lucent's network, the authors were able to find the attacker's subnet three out of five times, and were always able to determine the subnet within two to three hops.

The ethicality of this procedure was briefly touched upon, and it was acknowledged that this was of more academic interest – and should never be used in real situations on the Internet. Further, the program written to do the testing and analysis is not going to be released.

Analyzing Distributed Denial of Service Tools: The Shaft Case

Sven Dietrich, NASA Goddard Space Flight Center

The purpose of a denial of service (DoS) attack is to overwhelm the victim so that it is unresponsive to legitimate users, or to construct input to make the victim host act strangely or unreliably. The former is more common, general purpose, and what this paper focuses on.

The simplest form of DoS involves an attacking host sending packets directly to a victim host. As it was not always possible for a single host to saturate a victim, further refinements were made to DoS methods. Using amplifiers – hosts that amplify the amount of traffic output by an attacking host – became common with the "smurf" attack. Another possible method was to coordinate among multiple attackers to concentrate on a specific victim. However in the past, such coordination has been largely manual, such as agreeing to a time and victim through IRC.

Distributed DoS attacks are a recent "innovation" in the DoS toolkit. In the DDoS model, one or more attackers relay commands to a "handler" host, which maintains a list of "agents" – compromised machines running software to attack victim hosts. Through DDoS software, one individual can direct tens or hundreds of machines to attack targets, with the multiple sources of the attack making it highly effective and extremely difficult to stop.

The DDoS tool analyzed in-depth by the paper was called "shaft," and was the second such software available, after the original trinoo. Most analysis of shaft watched traffic between a compromised agent and a handler, as well as actual attacks which the agent participated in. Analysis revealed the attack methods the shaft tool used – a combination of TCP, UDP, and ICMP flooding – and the communication channel between the agent and handler was discovered.

SESSION: . . . DON'T LET THEM IN

Summarized by John Ouellette

YASSP! A Tool for Improving Solaris Security

Jean Chouanard, Xerox PARC

Purpose: to correct Solaris defaults:

- Permissive file modes
- Too many services running
- Inconsistent logging

Philosophy:

- One config file
- Model after Sun package – easy to uninstall
- Tolerant about what it expects
- Server or workstation based

YASSP's project:

- SECLean – core package
- Other: RCS, openssh, Tripwire, tcpd+rpcbind

Goal: to control startup of init scripts, while allowing the machine to return to its pre-YASSP state.

Needs: more English-friendly docs and testing for non-default Solaris systems.

Additional information on YASSP can be found at <*http://yassp.parc.xerox.com/*>.

NOOSE – Network Object-Oriented Security Examiner

Bruce Barnett, General Electric Corporate Research & Development

Purpose: to present tools that are lacking.

Content and function: distributed cooperative engine: Dispatcher, IW, GUI. There are presently 21,000 lines of Perl/Tk code at "research" quality level.

This tool checks for patches generated from the Sun FTP site; looks for Trojan horses; parses start files; tracks variables, $PATH, etc.; examines .rhosts; understands NIS netgroups; checks NFS access to users' homes.

Problems: single-threaded, not secure.

Performance: host with 2,000 accounts took 30 minutes to check 5,000 vulnerabilities.

Future: will be TCP based, multi-threaded.

Conclusion: object orientation is key to reusable algorithms.

Subdomain: Parsimonious Server Security

Crispin Cowan, Steve Beattie, Greg Kroah-Hartman, Calton Pu, Perry Wagle, and Virgil Gligor, WireX Communications, Inc.

Problem: granting least privilege not always easy, or feasible/possible. For example, mod_perl runs at Apache level of permissions.

Solution: ACLs for programs, instead of users. Subdomain is a kernel-level enhancement to confine programs.

For instance, program foo can be restricted by a config file:

```
foo {    /etc/readme, r
/etc/writeme, w }
```

This gives program foo read access to /etc/readme, and write access to /etc/writeme by invoking the chhat() method (i.e., change hat).

Workshop Series

Workshop 2: Teaching System Administration

Coordinators: Curt Freeland, University of Notre Dame, and John Sechrest, PEAK, Inc.

Summarized by Socrates Pichardo

This workshop was a continuation of last year's work. The goal this time was to brainstorm ideas for class material, exam

questions, concepts, etc. There was a good representation from all aspects of system administration education, with a heavy concentration of college/university educators. Here is the outline of the workshop:

Session 1
Concepts and prerequisites

Session 2
Concepts taught
War stories
Best exam question
What I wish class looked like
Getting students to participate in class
Measures of success
Questions for a prerequisite (minimum competency) exam

Session 3
Tools we could use
Develop examples

Session 4
Develop exam/homework/project
What we would like to have for a program/track/series/concentration
Workshop review

Results of this workshop will be published under the mailing list of the working group (<*sysadm-education@maillist.peak.org*>) and will be part, along with the Sysadm Taxonomy Project, Sysadm Certification Project, and Benchmarking and Measurement Project, of SAGE's efforts to formalize the system administration profession.

Workshop 3: MetaLISA

Coordinators: Cat Okita, Global Crossing, and Tom Limoncelli, Lumeta/Lucent Technologies

Summarized by Josh Simon

Six major topics were covered:

Staying Technical

We had a good hour-plus discussion on how to stay technical and manage your boss. Some subjects that came up included having both responsibility and authority, getting someone to do the nontechnical aspects while you concentrate on the technical ones, defining and reviewing roles and responsibilities, using agendas to run and control meetings, knowing when to say "I don't know," holding regular "town hall" type meetings for the user community, and apologizing when you screw something up.

Retention, Hiring, and "Coaching Out"

The general consensus here was to involve the Legal and Human Resources departments as soon as possible and to document everything. If you need to encourage someone to leave, you can either treat it as a pure performance issue or possibly a security issue (for example, if the employee in question has root). If performance is an issue, you can use improvement plans or a probationary period. You'll definitely have to manage the morale of those who stay.

This led to a discussion on hiring. First, how do you find qualified applicants? You can look online, though for more signal and less noise you can go by word of mouth, the SAGE jobs list, and even campus recruiters. Second, how do you convince these qualified folks to join your company? Some thoughts for managers focused on looking at the long term, not the short term, since you cannot easily get rid of someone you've hired: do you have to raise or adjust the salary of the new hire? of the rest of the team? do you have to train people? Are bonuses involved?

This led to a discussion on retention. In order to keep employees on your team, we determined that managers need to have flexibility in providing raises (both in terms of frequency and amount) and reviews (more often than just annually). Providing perks, such as training, conferences, laptops, high-speed network access, soda and beer, flex time, toys to play with (both computer-related and non-), cool projects, good management, good co-workers, good environment, and respect and recognition can help you retain your best employees.

Leaving Gracefully

If you find yourself in the position of leaving a job, you should leave gracefully. Hand off your responsibilities, make sure no batch or cron jobs run from your own account, document everything (both what happens and why), and put read-me files in nonstandard directories and hosts. Be professional and do what's right for the company; don't send any hate mail. Whenever possible you should train your replacement. As a manager, you need to plan for your people leaving, be it by leaving the department, leaving the company, or being promoted out of a position. Some other questions that arose included:

- How do you decide when to quit?
- How do you handle a subordinate being promoted over you?
- How do you manage friends?

Talking (Up) to Management

When talking to your management you have to remember to tune to the audience. Talk about the technical issue in terms that your audience is familiar with. You need to focus on the business reasons and issues and not the technical jargon. You should ask your peers, or even your boss, to review any messages you're about to send. Find something in common with the manager and use that as a basis for establishing rapport in your communications.

Talking to Now-Subordinate Peers

When you're the one who's been promoted to lead your team, there are a few things you need to remember. You have to be careful with social events; as a manager you're no longer "one of the gang." There is going to be some information you cannot share with your team, and they are going to know that. You have to treat everyone the same regardless of how you may feel toward them (such as friends and non-friends in the same team). The dynamics will differ by group

size; managing one or two people is different from managing 10 or 12. Finally, you have to be objective and impartial.

General Tips and Techniques

We wrapped up the day with some general tips and techniques for being good managers.

- Set boundaries for your team.
- Let your employees fail.
- Remember to test, and include testing in projects.
- Consider when to delegate or take over something.
- Review your team, your peers, and your management.
- Spend a lot of time up front on concepts and requirements of projects.
- Protect or insulate or buffer the team from more-senior management.
- Back up (support) your employees; trust your team.
- Stay out of the office to (1) delegate and (2) find out what does and doesn't work without you.
- Don't micro-manage your employees.
- Communicate up, down, and across; open communications are very important.
- Teach skills, not things or details.
- Don't lie to yourself.
- Assume the other party is trying to do what's right for the company.
- Bounce thoughts off peer-level managers.
- Find a mentor (either inside or outside your organization).
- Do one thing every day that scares you.
- Match customer expectations with reality; prioritize.
- Rotate your team through the various positions to reduce the risk of burnout.
- Don't decree decisions, unless you have no time to reach consensus, you cannot reach a consensus, or there is an obvious violation of policy.
- Kill off or postpone projects when necessary.

Workshop 5: Advanced Topics

Coordinator: Adam Moskowitz, LION Bioscience Research, Inc.

Summarized by Josh Simon

The professionalization of systems administration was one of the topics discussed. A comparison was made to doctors. We use similar skill sets – diagnosis, comparability, problem solving, and so on. But can it be said that lives are at stake when systems administrators do their job? Doctors charge by the visit or the procedure; systems administrators don't. The models are, however, converging in some ways. Many systems administrators are more concerned with architecture than are doctors. There are differences in scale: doctors are like help desks, while systems administrators tend to serve larger numbers of people. Doctors are, in fact, certified. Some systems administrators contravene organizational policies. Doctors are liable, lawyers are liable, and engineers are liable; systems administrators are rarely liable. This led to a discussion on professions: professions have standards for training and knowledge (certification); there's a fixed set of information. Sysadmins are often grassroots with self-training and apprenticeship. Certification is a required stepping stone. Maybe systems administration should be a "guild." Or maybe we should form a union.

A second area of discussion was whether or not ISPs are now perceived as commodities and whether they can be run as commodities. The consensus was that they can, but you should be sure to check out their long-term business prospects because business models change rapidly. Finding a provider for services "beyond the basics" is hard. ISP consolidation is in progress. Any new ISP will require new technology. Not only are ISPs perceived to be commodities, so are their users (who are traded). Local and national ISPs can survive; but it's tough for regional ISPs, who are neither local nor a brand name. Are there brokers for customers? There are special deals among ISPs, but no B2B site. DSL was enabled by aggregating terminations at the central office. Those who can scale will survive. You can now purchase a turn-key 10- to 50K-user ISP solution that requires very low levels of sysadmin skill. Shell accounts are a thing of the past; people are running their own servers at the end of a DSL line.

A third major area of discussion was on separating policy and implementation. One possible solution is to have an interpreted "policy language." Maybe you can use general principles and then color the bottom-level implementation to match existing policies. This is more of a mindset problem than a coding problem. Let's build policy engines, not engineer accounting (or whatever) systems. Cfengine has features that can help you implement policies. You must codify the policy in a way that's measurable so you know if you're "on policy" or not (and then you can get back on policy if you get far enough "off track"). We're already adapting host-based tools that query directories. Maybe we can graft policy engines onto directory responses.

A fourth area of discussion was on how new technologies in the last few years seem to be languages. This is true because languages can express extensible ideas; they build from primitives and move to greater complexity. Some people say "use a database for policy," but databases too often require predefinitions. Languages, on the other hand, are infinitely extensible. We think this is the solution for policy expression. A well-crafted language could potentially address this problem, but we don't know of one right now. We think languages can express these specifications at the proper level. The real problem is the ability to describe when a particular operation is authorized. We need to agitate for rich-

ness of expression in commercial tools. Windows has a lot of configurable options under the hood that were difficult to access via the desktop or command line, even though an API was available. Declarative languages like Prolog might be able to help here. Exceptions are surely the difficult and important part of this problem.

We wrapped up by looking at our 1998 and 1999 predictions to see if we were late or still wrong. We still have more misses than hits. In 1999, 9 of our 19 predictions came true (or mostly true), for a 47% success rate. More of our 1998 predictions came true in 2000, but we're still looking at about a 50% success rate.

Our predictions for 2001 are:

- Peer to peer (including systems administration) will grow, then shrink. (85%)
- DSL-based ISPs will grow in popularity, then die as the tech-savvy turn their DSL to a friends-and-family ISP. (65%)
- Alternate dialtone-to-home competition will break loose (50% of the market) this year – telephone companies will have to change their business plans. (100%)
- The number of purported PDA- and managed home-appliance systems will double in the next 12 months. (75%)
- Some time this Christmas season things will go well with e-commerce. (100%)
- There'll be an e-commerce disaster some time in the next year, though. Or, a Fortune 100 company will have an above-the-fold e-disaster. (40% think it'll show up in print)
- We will have at least one Microsoft-facilitated security bug à la Melissa or ILOVEYOU in the next year. (100%)
- There'll be at least one major Silicon Valley power outage that provides above-the-fold problems for at least one company. (85%)
- Many dot-coms with otherwise profitable, viable business models will fail because their names aren't AOL or Yahoo (investor confidence will drop further). (70%)
- This is NOT the year that Silicon Valley loses its shine and people start a mass exodus. (75%)
- 802.11b will become standard on all business desktops and laptops in the next year. (80% on desktops, 100% on laptops)
- 802.11b public Internet access will be available in the top 25 US airports by December 2001. (100%)
- A huge mobile phone will be dug up on the surface of the moon.
- You will not see networks on airplanes in 2001 (not counting dial-out via airphone). (100%)
- Businesses will find their storage (at least) doubling this year, with the concomitant backup problems. (75%)
- Linux will splinter (speciate). (10%)
- The price of 17-inch flat panels will come down to $700. (100%)
- 200-dpi-resolution displays will appear on desktops. (10%)
- Gigabit Ethernet hubs will dramatically decline in price in 2001. (35%)
- Serialization of Object Application Protocol (SOAP) – RPC over HTTP; will ascend to wide acceptance. (15%)
- Official or commercial music delivery services will fail. (85%)

Finally, we listed some cool tools we're using:

- VMware
- Rethinking the world in terms of PHP, MySQL, and HTTP
- Wireless everything (802.11b)
- Some of the new load-management tools (batch queuing tools) for clusters
- 65-pound brute-breaker demolition hammer with its own cart
- PL/SQL
- XML and JavaScript
- Wiki (a very simple browser-based editing environment)
- VNC (virtual network consoles for NT)
- Baytech power strips with Ethernet access to remotely reboot via power outlet (vector for the Microsoft security outage we're predicting)
- Blackberry (two-way) pager
- Unison (file synchronizer tool between desktop and laptop, two-way comparison, etc.)
- Python
- Palm/Handspring
- ssh in IOS
- Netflow (Cisco-created protocol gaining acceptance) tools that've come out in the past year; you can now do accounting without trashing performance
- tangram (<*http://www.tangram.org*>); a Perl module that makes variables to persistent storage (SQL database)
- Herman-Miller Aeron chair; it's really worth it if you sit on your ass all day
- Authenticated Web environment that keeps the authentication token on all the time
- Win32's NetCaptor; tabbed Web-browsing Web interface
- Win32's PowerMarks; bookmark manager that has keyword-based searches
- blog (short for weblog) that logs where you go and lets you store a bunch of stuff in a column as if you were writing a letter or column like slashdot; similar to userland (see <*http://www.userland.com*> for details). Can be published as RSS feeds to JavaScript news-ticker applications.
- Newsbytes-style column

Workshop 8: System Administration Process Improvement, Benchmarking, and Metrics

Coordinators: Carolyn Hennings, Megapipe, Tom Limoncelli, Lumeta/ Lucent Technologies, and Alva L. Couch, Tufts University

Summarized by Nicholas Schrieber

Caveat: this report is not intended to be minutes of the meeting.

Some workshops are odd, especially the first one on a topic, with no clear agenda or mission. The mission options at the start range from disbanding at the end of the day, saying, "OK, we've exhausted that topic," all the way to planting the seed of what eventually becomes a 500-member consortium with an executive director.

The eight of us who attended spent much of the day exploring an appropriate mission for the group. We examined and compared possible forms that a useful SA process improvement program or document could take, and we tried to determine appropriate roles for the committee within SAGE, and vice-versa. The initial mission statement we decided on for our day's work was: "Develop a framework and methodology for measuring and improving organizational maturity with respect to SA practices." At the end of the day, it was very clear that we had not actually even begun a framework and methodology, but we had reached some decisions on the shape it should take, as well as the role it should play vis-à-vis SAGE and several other SAGE-recognized programs.

Among the various models we had to consider were the System Administration Maturity Model (1993), by Carol Kubicki, the SEI Software Capability Maturity Model upon which it was based, some high-level process measurement techniques a committee member presented, as well as the Systems Administration Body of Knowledge, for which Geoff Halprin is preparing a Guide document. We also needed to consider the SAGE taxonomy project and appropriate complementarity with it.

We need to be able to ensure reproducible results, not just repeatability of process. This distinction represents an easy shortcoming many such models could harbor. Those of us who had read Carol's 1993 SAMM were in consensus that it was a good solid attempt at what was needed. There were two problems we saw, one being the state of the systems administration industry at that time, and the other is that, even now, the model as she presented it is probably not clear enough to make its widespread adoption likely. We seemed to agree that the state of the industry has advanced, and even the codification of the in-progress SA BOK suggests we're now ready to apply Carol's SAMM principles. But they need to be clearer, less academic, and probably should incorporate some changes that additional years and a larger committee can bring to the project.

The final outcome of the day was multipart:

- We would work toward the development of a new SAMM document that is essentially reflective of the 1993 document, but greatly improved.
- We would ask SAGE to officially sponsor the project, which would in itself facilitate the status of the document toward becoming a standard. We consider that this sponsorship is already basically a fact, but requires clarification. Perhaps the best term is "SAGE-recognized" rather than sponsored.
- Further, we would ask SAGE for funding that might be necessary to provide staffing for further development of the new SAMM document.
- Carolyn Hennings would write a formal proposal, probably for presentation at the February SAGE board meeting.

Other loose notes from the day include:

Part of this job is the creation of a metrics language. This obviously dovetails very significantly with the taxonomy project. We need to take a step back and internalize existing taxonomy.

Alva Couch presented some interesting ideas on metric (measurement) systems. He pointed out that the biggest problem is too many variables.

As a high-level summary, he made reference to the Boehm approach from SE and proposed a modal approach. A major aim would be to relate life cycle cost over ideal cost (complexity of task). His conclusions:

- Robustness: ability to survive changes = cost of replacement/complexity of assigned tasks
- Efficiency: relative cost of life cycle to ideal

Over the long run the group would have to deal with complex couplings of major issues. For example, coupling with the SA BOK project is on the one hand obvious and perhaps inevitable. But it could raise false presumptions about their compatibility or the "ownership" of the SA guide project. Geoff Halprin is currently the author and owner, although he expects to allow free use of it, as long as his authorship is respected. In the Project Management field, the PMI owns the guide to the PM BOK, and as such it does not act as a private person or corporation, but more as a standards group.

Workshop 9: Taxonomy

Coordinator: Rob Kolstad

Summarized by Dave Bianchi

The goal of the workshop was to create a framework for a taxonomy of systems management to enumerate all the tasks that a systems manager might perform.

The eventual goal of the taxonomy project is to provide a list of tasks that a system manager performs along with a short description of each task. Rob would

like to have a fairly complete list of tasks within three months.

The first half of the day was a brainstorming session, naming tasks and trying to group them in some meaningful way. The group came up with a grid system to categorize common activities around a task. Common activities included policy, standards, security, budgeting, and legal issues.

The second half of the day was spent focusing on just one task: "Printers." Rob proposed that a Web site be created to allow a group of people to work on the rest of the tasks, and he volunteered to create the Web pages. The group volunteered to help maintain the Web pages over the next three months.

See: <*http://ace.delos.com/taxongate*>.

BoFs

FreeBSD

Summarized by Eric Lakin

The FreeBSD BoF was chaired by two FreeBSD developers, one of whom is a core member (David Greenman). Questions were mostly directed at the two developers, with only minor help from the audience. For some questions, this was appropriate – such as the current "state" of FreeBSD and Core, and the progress of BSDi integration into FreeBSD.

"Core" is a group of developers that oversees the direction of FreeBSD development. Until recently, a person joined the core by developing and proving their ability, and then being asked to join. In the past year, there was a first-ever election of the core (by the active developers, I believe).

Expectations for the integration of BSDi/FreeBSD following the merger of Walnut Creek CDROM and BSDi have failed to materialize. A rewrite of the SMP sections of the kernel are currently in progress, with design influences from the BSD/OS code, but no actual code mingling has occurred yet.

Of particular interest was the discussion of a logging or journaled file system in FreeBSD; on-hand in the audience was Kirk McKusick, the architect of the BSD Fast File System. He gave an overview of SoftUpdates, an FFS addition currently available in FreeBSD that increases performance and addresses some of the issues that relate specifically to journaling file systems. Theoretically, at least, fscking a file system shouldn't be necessary when softupdates are enabled, but fscking is currently still done "just in case" until the code and theory have proven themselves.

The remaining discussions mostly related to individual problems with FreeBSD, most of which boiled down to "more information needed."

Getting Plugged Into Sendmail

Mike Smith, ActiveState

Summarized by Theo van Dinter

This BoF was actually titled "libmilter: Getting Plugged Into Sendmail." Libmilter is included in Sendmail 8.10 and above, and gives access to the Sendmail Mail Filter (milter) API. Milter gives hooks into every stage of an SMTP transaction, and lets you perform custom actions based on any part of said transaction, including the content of the message. While some of this is possible without milter (for example, procmail lets you filter after message acceptance), milter gives you more functionality before the message has been accepted by the mail server.

Some things you can do with milter:

- **Mail archiving:** selectively archive messages based on specific criteria
- **Spam control:** deny mail delivery based on your programmed specifications
- **Content rewriting:** add new headers or change existing content
- **Virus scanning:** verify that incoming/outgoing attachments are clean

All of the examples shown during the BoF were written using ActiveState's PerlMX product, which allows these filters to be written in Perl instead of the usual C. It is a commercial product but is available for free to individuals and educational sites. For more information, see <*http://www.activestate.com/PerlMX/*> and libmilter/README from the Sendmail distribution.

BoF: NetBackup

Curtis Preston, Collective Technologies

Summarized by Steve Hanson

Due to some scheduling confusion, Preston was late coming to this BoF, but in normal USENIX fashion the attendees (the room was very full) proceeded to run the BoF themselves, and a lively discussion of various NetBackup technical and support issues began. This mostly revolved around the typical vendor complaints and a few specific issues with backup scheduling and system security in NetBackup.

After Curtis arrived, the discussion quickly became more of a tutorial, which was quite interesting. Most of the information was on `bpgp`, an undocumented command in NetBackup which is used internally to copy files between systems. Although one could consider `bpgp` to be a security hole (at least on NetBackup systems which are not using the built-in authenticated communication methods), it is a very useful tool for copying configuration files and other information between systems which are NetBackup clients or servers. Some typical uses include dissemination of exclude files and any other sort of configuration information. In all, this was a good and worthwhile session, though it could have been much more effective if it had been scheduled for longer than an hour.

Postmaster

Strata Rose Chalup, VirtualNet Consulting

Summarized by Theo van Dinter

This BoF was intended to get people together to talk about SAGE's upcoming "Role of Postmaster" booklet. The booklet is meant as a "best-practices" guide for system administrators in charge of mail systems, but will not be limited to any particular mail transport agent (MTA). The booklet is being co-authored by Strata Rose Chalup and Brian Kirouac. There were far too many suggestions and planned topics to be listed here. This booklet might actually have to be split into multiple booklets to cover all of the different parts of the postmaster role. If you're interested in becoming involved with the booklet (either by making submissions/suggestions or just seeing what everyone else is sending in), please contact Strata via email: <*strata@virtual.net*>.

Sendmail – Open Source / Commercial

Summarized by Brian Kirouac

Open Source 8.12:

- new IO library
- new memory management to limit forks
- no longer need to get sfio for security add-ons

Open Source 9.x:

- global optimization
- some things from qmail and postfix
- not a monolithic program, but will not go as far as postfix ("postfix has too many small processes")
- most processes will be threaded
- some platforms will be dropped
- new configuration files
- ambitious goal for performance: machines two or three generations out will be able to handle 100 million messages a day, and this will work on clusters

SMI Advance Messaging Server:

- release 2.5weeks ago

SMI:

- Eric is being cautious
- he hasn't killed the new CEO yet
- still in the process of going through ripples of new CEO
- "still going wonderfully"

The last CERT advisory for Sendmail was in January of 1997.

LISA 2000 Solaris Docs

Summarized by Steve Hanson

The Solaris Docs BoF was held by several of the Sun documentation developers. The primary purpose of the BoF was to solicit input from Solaris Documentation users and to answer questions they posed.

Several issues were raised during the BoF, including:

- Jumpstart documentation is out of date. Attempts are being made to improve it for Solaris 9.
- There were several complaints about the navigation on sun.com in general, and on docs.sun.com in particular.
- Several users wanted documentation for End-of-Life'd products kept online so that users of older hardware can still obtain information on their products.
- One of the primary improvements made in the last several documentation releases was to try to include more task-specific information. This was prompted by user requests.

There should be several upcoming improvements in the BigAdmin site at Sun, including tying BigAdmin into the other documentation sites and providing more information on upcoming Solaris releases. BigAdmin is a useful Solaris administration resource and is located at <*http://www.sun.com/bigadmin*>.

Email comments on documentation can be sent via the comment alias on docs.sun.com, or by sending directly to the documentation developers:

<*cindy.swearingen@sun.com*>
<*julie.nelson@sun.com*>
<*kathy.slattery@sun.com*>

UNIX on Handhelds

Summarized by Steve Wormley

Steve Wormley led a BoF on "UNIX on Handhelds and Wearables" Wednesday evening. It turned out to be mostly on wearables by a guy from MIT (next time, just handhelds will be covered).

It was an interesting group with lots of questions and details of what is being done in the field. The head-mounted displays are getting smaller, the networking (802.11 and CDPD) is getting more widespread, and the devices are getting more powerful.

All in all, it was interesting but a bit short on handhelds, and they definitely still aren't for everyone, and not even for most of us yet.

conference reports

COOTS 6

by Peter H. Salus
Our peripatetic historian
<peter@matrix.net>

How many conferences/symposia/workshops of type X are enough conferences/workshops of type X?

I got the feeling at COOTS 6 (6th USENIX Conference on Object-Oriented Technologies and Systems) in San Antonio in January that nearly 100 of us loyalists were determined to answer that riddle. Is that a conference or a workshop?

There were five USENIX Graphics Workshops, but there hasn't been one in over a decade. Tcl/Tk seems to have run its course. Etc.

Most of us use objects. Object-oriented technology is foreign to no one in the community. But how many workshops does one need? I don't really know. I enjoyed this one a lot; it was small enough that I was able to actually say hello to nearly everyone I wanted to. For the first time since Toronto, I didn't ask to be invited to the Advanced Topics Workshop.

I got a chance to talk to Ken Arnold and Deborah Zukowski, to Doug Lea and Rajendra Raj. And, once more, to hear the invariably entertaining Stu Feldman.

Stu gave the best keynote (in my opinion) at any USENIX conference – in Salt Lake City in the distant past (1984). He was the keynote at another USENIX in 1992 in San Antonio, too.

In SLC his talk was disrupted when the hi-tech slide projector failed; in San Antonio, there was a fire alarm. Stu survived.

Feldman's thrust was on two things: speed of change ("every six months a new gadget") and ubiquity. The tighter the software community, the worse the interface constraints. Do we want to be constrained to the brief messages of Japanese schoolgirls? He pointed out that there was a $300 billion investment in mobile electronic business.

In an historic aside, Stu pointed out that we had moved from the mammoth centralized server to the client server, to the Internet, and that we were becoming Web/Network-centric. This means that personalization, notification, efficiency of information, and location sensitivity are coming. He pointed out that Dick Tracy's wristwatch was already in experimental form and might well be available soon. (But, he admitted, we'd have to wear a battery pack.)

We will be operating with "layers of service" and "nets of information." There is thus no simple device model. We've been living with the PC model, but we're moving further into diversity. This will be dreadfully exciting; we'll deal with peta-everything. We need high connectivity, device capability, and integrated services, says Feldman. The problems will lie in the areas of authentication, authorization, and transfer.

"Devices are objects. User models are objects. Business models are objects. Business processes are affected by the changing state of such objects."

"Applications are the real driver," Feldman concluded.

It was a really fine talk; lots to think about. Thanks, Stu.

That afternoon, there was a guest lecture by Robert Martin of Extreme Programming. If you know nothing about XP, I suggest one of the Addison-Wesley books. Martin defined XP as "a set of simple rules from which complex behaviors arise." His form of this might be summarized as:

Due Date is #1
Due Date is frozen
The spec is never frozen (the specs change all the time)
Ad hoc management
Waterfall model

Analysis/Design/Implementation go ahead in parallel (with feedback), not sequentially.

Martin made a number of really good points:

- If you reduce quality, you slow down
- Dates can't be changed
- Adding staff makes it later (Brooks)
- You can change scope by deleting features

He also offered some "rules":

- Write tests before code
- Program in pairs
- Integrate frequently
- Rest
- Communicate with customers daily
- Follow customers' priorities
- Leave software clean/simple at the end of the day
- Adjust processes and practices to your environment

Martin was entertaining and provocative.

The next morning, Bjorn Freeman-Benson spoke about software management perspectives. Using the analogy of orienteering, he spoke about several large software projects he had worked on and their "problems." OTI, Rational, Amazon.com, and QuickSilver Tech all had different problems.

OTI built a "repository-based solution, not a file-based solution." VisualAge was what "we wanted to build, not what the customers wanted." Rational wanted to create a totally new system, not a build on top of other stuff – not really a "legacy system."

Amazon built many little programs in C, rather than objects. This made training and refactoring expensive, though maintenance costs were low. Amazon ended up with three distinct layers: db (Oracle), templates (internal scripting language, obidos, and Perl), and software (plugins to Apache, in C/C++).

Quicksilver has concentrated on using lightweight processes to adapt to business needs.

Freeman-Benson's point was: "Engineering is about the big picture *and* about the details": set a goal, plan, execute, be flexible.

Once upon a time, a decade ago and more, Smalltalk, C++, Eiffel, Java were new or nascent. COOTS is now more about the big problems than the little ones. The Stroustrup-Cargill discussions about inheritance are of little import. Ken Arnold's BoF on Jini was a big deal.

With under 100 attendees, maybe COOTS should now be shelved; something else might sprout up.

I'm pleased it's not my decision.

needles in the craystack: when machines get sick

Part 3: The Myth of Computer Control

by Mark Burgess

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

<Mark.Burgess@iu.hioslo.no>

If apathy could place our technological future in jeopardy, making us slaves rather than masters of our contrivances, then the diametric pitfall is a simple arrogance: the desire to control it all. The absolute control of computers is doomed to failure, for the same reasons that the absolute control of weather and environment are doomed to failure. Such complexity cannot be controlled, it can only be regulated. To understand why and what this means for the administration of our systems, we must leave the relative safety of simplification, and dive into a more turbulent sea, in search of detail.

Imagine opening up a computer and aiming a microscope inside. Imagine turning up the magnification step by step through circuit boards, components, materials, and, finally, atoms and electrons. Clearly the important electrical behavior of a computer takes place at the level of electrons, so we could in principle understand almost everything about a computer in terms of (1) the arrangement of its atoms and (2) the motions of its electrons. Although possible in principle we could never do it in practice. We would never be able to cope with all those details in one go. It would be like trying to explain the trends and goings-on in a city by tracing the movements and interactions of each of its inhabitants over time.

Our brains have limited processing power; we are not good at modeling complex details and extracting long-term trends, so we must use a strategy of divide and conquer, *by scale*. "Scaling" is a common term in physics, where it has long been understood that an understanding of complex phenomena can, in many cases, be separated into the understanding of different partial-phenomena at different scales or sizes. This was how the magician's illusion worked in Part 2 of this series.

If we return to the microscope analogy, it seems reasonable to divide the understanding of computers up into scales, which minimize the number of independent levels compounded to create the illusory magic of computers. For instance, there is the level at which electrons move around electric circuits in controlled ways, leading to microscopic switching technology. The transistor, invented by twice Nobel laureate John Bardeen (his second prize was awarded jointly with Leon Cooper and Robert Schrieffer for the BCS theory of superconductivity), was the key development that paved the way to miniaturization of digital technology. Then there is the level at which transistors are combined to make Boolean logic gates, the level at which logic gates are combined to produce processors and memory, then the level at which these are combined into a working computer with devices attached, the level at which computers are combined into networks, and so on.

These levels are essentially independent. Each scale adds new information, which does not depend significantly on the details of how the lower levels worked. So, it would not matter to the construction of computers if one replaced transistors with optical switches, the operation of the higher levels would be the same. It might be faster or heavier, so specifics might vary, but the evaluation of answers would not be affected by the changes. This is fortunate, otherwise we would have to redesign computers every time a new switching technology came about. Systems are routinely designed in layers

so that components can be exchanged for newer and better ones, without causing turmoil. For instance, think of the OSI network model, or TCP/IP. Telnet works over serial lines, or via TCP/IP over Ethernet, or with TCP/IP over fiber optics, etc. It is like procedural programming. It is the most important principle in computing.

This principle is applicable in many situations in science. For instance, one does not need to know about quarks in order to describe planetary motion, even though planets are almost entirely made up of quarks. The scale at which quark interactions are important is so small that the results do not depend on what the quarks do to a hundred decimal places or more. In other words, low-level differences can be ignored to a very good approximation.

The Most Important Idea in Computing

Paradoxically, we tend to claim that we understand something if we know how to express it in terms of smaller parts. This idea is the essence of all hierarchical thinking: directory structure in file systems, process trees, and so on. It is all about hiding unnecessary detail in Pandora boxes, to simplify and thereby reveal clarity of structure. As long as we know how to open a black box, we are happy to say that we understand it, at least in principle.

Paradoxically, we tend to claim that we understand something if we know how to express it in terms of smaller parts.

Of course, this reduction is not always possible: there are prominent counterexamples. Quantum mechanics is one: it cannot be reduced into anything else we know about. Quantum mechanics consists of a set of rules for calculating the mechanics of particles at the microscopic level, which works with extraordinary precision. Despite this, no one knows how to explain these rules in terms of anything deeper, and for that reason most of us feel that we do not understand quantum mechanics, and rightly so. Another example is nonlinear systems where chaos leads to a mixing of levels, in often unpredictable ways.

But how well does anyone understand anything? The illusion of understanding is, in a sense, proportional to the number of times we can reduce a thing into smaller or more fundamental things. At some point that process has to stop, and we end up with pieces which we can no longer break down. So how deep do we need to go in order to claim to understand something? In fact, most of us are not very demanding; if we were, we would probably not be able to cope with the detail required.

The principle of reductionism is simple to state: an efficient way to understand any *thing* is to take it apart, breaking it down into its constituent parts to see how these parts interact to form the whole. Usually this is a complex undertaking, and we never manage to complete this program of inquiry. Often the parts themselves are so numerous and so unexpected that their study tends to dominate the discussion, and the important information about how they fit together is left out. This has certainly been true of system administration.

If we want to understand a car, we begin by describing its main features and then proceed by taking it apart and examining each of those parts. Each part which we remove can be broken down into smaller and smaller parts until, in the end, we come down to atoms and their constituents. The basic Lego pieces. We learn a lot by this reductionist process, but it is crucial to understand that, at every stage of the reduction procedure, we *lose* the information about how to put the parts back together in the right places in the right order – that is, the information at the level above. In the end, the Lego bricks all look the same and tell us little about the original construction.

Understanding the relationship between continuous and digital signals is a step toward understanding the implications of complexity for computer systems.

It is the structural information which distinguishes a car from a refrigerator, or a horse from a block of buildings. When it comes down to it everything is made up of a few basic constituents. What defines unique entities is the information about how to put them together. Once again, it is about levels of details, where structure does not necessarily depend on the exact nature of the building blocks, to some degree of approximation.

Analogy, Approximation, and Cognition

So is this about approximation? In a sense it is. Computers were not always digital by construction. Before digital electronic computers were built, electrical circuits were constructed to model particular calculations with voltages and currents. These continuous signals evaluated continuous solutions to differential equations. They did not work to a fixed digit floating point accuracy, they worked by *analogy* and answers were obtained with voltmeters and galvanometers. They were thus referred to as analog computers, or analogy engines. The word "analog" has since entered our vernacular with the meaning "opposite of digital," which is clearly nothing to do with its root.

Understanding the relationship between continuous and digital signals is a step toward understanding the implications of complexity for computer systems. Everything looks more blurry than it is. There is an inherent uncertainty, not only in our senses, but in any measurement, due to the limits of digitization (the resolution) of a scale of measurement. The same uncertainty lies in any digital or digitized process. There are no truly continuous signals. Even detailed sources are only sampled with limited resolution, so their original nature is neither relevant nor determinable.

When we play a CD we hear continuous music, not digital staccato. The resolution of audio sampling on a CD is greater than the sampling rate of our ears, and thus it is beyond our ability to discern the difference: higher Fourier modes that might otherwise unmask the charade are not available to our perceptual apparatus and thus we are incapable of knowing more without artificial aids. Even artificial aids have limitations. They might extend the ability to detect greater resolution, but no instrument or physical object can provide infinite detail, for the simple reason that the universe itself has finite detail.

A CD recording is but a simulacrum of an original physical (musical) process: wasn't that original process continuous, with infinite detail? (Hi-fi buffs have sometimes used this argument in favor of LPs over CDs.) The answer is no: the bow of the violin is a rough surface which plucks the string very fast, giving only the illusion of continuous play. The grains in the vinyl of an LP imply a maximum resolution that can be represented in that medium, beyond which is unpredictable noise. At some point (for whatever reason) one comes down to a minimum size for representing information: single atoms, for instance, probably represent the smallest units one can use to represent information with any fidelity.

Our sensory-cognitive apparatus has a great influence on the way we go about analyzing information and "understanding" it. So much so that its prejudices are frequently fooled by optical illusions, Rorschach tests, blurry images, color blindness, Martian canals, etc., etc. Our desire to see patterns and to fill in gaps, coupled with the experience that a closer look reveals more detail, builds a tower of assumptions about how things behave. We are seldom asked, by nature, to examine the limits or consequences of digitization, but when analyzing scientifically, we are obliged to explore these consequences.

As humans, we think and assign meaning in digital terms: either we coarsely classify shades (blue is still blue, no matter how light or dark, turquoise is still blue-green, no matter what mixture), or we enumerate classes (like red, green, blue, with arbitrary shade boundaries). In spite of this, we still maintain the illusion of continuous change, and we describe it mathematically as the limit of a digital process, abhorring the discontinuous in favor of smoothly changing lines.

Continuous representation of information is a thorn in the side for pattern recognition. True pattern recognition can only be achieved for a digital representation and the same is therefore true of computer control, since the elements of control are digital instructions. When two strings of digits are the same, a match is found. We can match any kind of blue, only because the digital concept of blue represents a whole range of colors. But the range of possible blues means that there is an uncertainty in the meaning of "blue." It is a class of finite width. When a digitization is very detailed, i.e., approaching the continuous, it is not always practically possible to distinguish every digit of data. Similarly, it is not practical to issue very complex instructions at the microscopic level.

Analog and Digital: Some Information Theory

This makes more sense if one knows some information theory. The study of information began in the 1930s and 1940s by mathematicians such as Church, Turing, Von Neumann, and Shannon. What we call information theory today was largely worked out by Claude Shannon (of Bell Labs) and published in 1949 under the title *The Mathematical Theory Of Communication*. He defined the mathematical measure of information (entropy) and devised many of the core theorems used in information theory today.

Information theory is about the *representation* (manner of storage), *interpretation*, and *transmission* of patterns of data, i.e., patterns made up of different kinds of "something" and "nothing." We attach meaning to these patterns and call the result "information." Patterns of data are mainly of interest when they are transmitted from a *source* to a *receiver*: e.g.,

- Text read from page to brain
- Morse code sent by telegraph or by lantern
- Speech transmitted acoustically or by telephony
- Copy data from hard-disk to memory
- Copy data from memory to screen
- Copy DNA using RNA within a cell

In each case, a pattern is transferred from one representation to another. Information is fundamentally about representation. It is the limitations implied by a given representation which form the substance of information theory. Some of the issues one has to contend with are:

- Distinguishing patterns of "something" and "nothing"
- Perhaps distinguishing different kinds of "something" (classification)
- Space-time coordinates and units (is a long beep the same as a short beep? quantization/digitization)
- The meaning of redundancy and repetition in a signal

To build a more precise picture of what information is, we can begin with a signal f(t,x,..), of unknown detail, which is a function or field that changes in time or space. The signal is really just a pattern formed by a disturbance in a physical medium. Our everyday experience leads us to believe that there are two types of signal f(s):

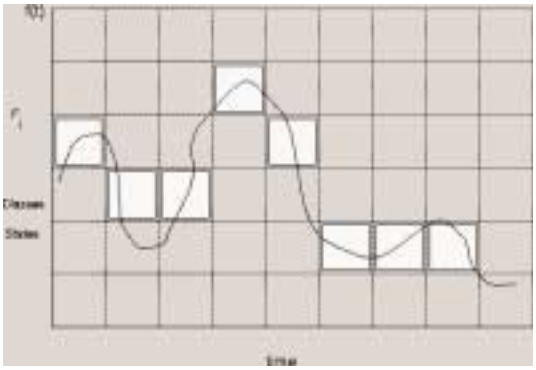


Fig 1. Coarse-graining or digitization is a coordination of the continuous signal.

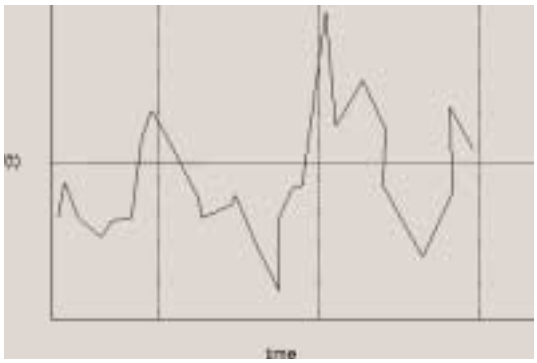


Fig. 2. A poor digitization cannot sensibly determine the value of the signal within the cells.

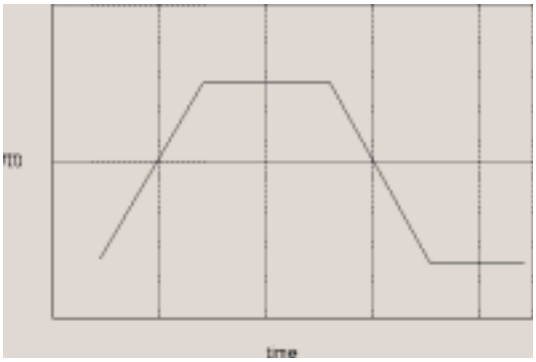


Fig. 3. A well-suited digitization without loss. This signal can be represented with just two classes, i.e., binary digitization.

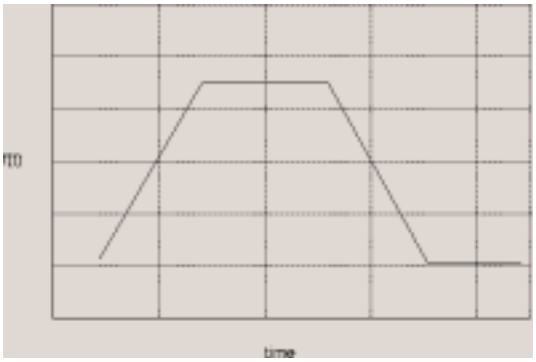


Fig. 4. The same signal, as in Figure 3, this time digitized into six classes.

- Analog or continuous functions f(t)
- Digital or discontinuous (step) functions

It should already be clear that this distinction is artificial but that in this distinction lies the central essence of what information is about. An analog signal is a limiting case of a digital signal.

In order to say anything about the signal, one has to map it out by placing it on a discrete grid of coordinates. At some arbitrary level, one decides not to subdivide space-time any further and one reaches a limit of resolution.

Information can only be defined relative to such a set of coordinates since it is the only means one has of describing change. Let us begin by assuming that the detail in the signal is greater than the resolution of the grid. We do the following:

- Divide up the time axis into steps of equal width dt. Here we shall look at an interval of time from t=0 to t=N dt, for some N.
- Divide up the f-axis into C classes $F_i = [F_i^-, F_i^+]$, which touch such that $F_i^+ = F_{i+1}^-$.

Digitization means that, whenever the function f(t) is mostly inside a cell F_i, its value is simply represented by the cell. We have compressed the detail in the square region F_i into a single *representative* value i, over an interval of time.

There are good digitizations and bad digitizations (see Figures 2, 3, 4). Nyquist's law tells us that the interval widths need to be half the width of the "finest change" in the signal. In Fourier language, the sampling rate must be twice that of the greatest frequency we wish to resolve. This is why CD players sample at 44.1kHz and DAT samples at 48kHz: the limit of human hearing is about 20kHz when we are young, and falls off to about 12kHz as we grow old.

The digits F_i are the "atomic" units (the Lego bricks) of information: they are a strict model for representing change. If C=2, we have binary digits $\{F_1, F_2\} = \{0,1\}$ etc., or *bits*.

God Is a Pile of Bricks but the Devil Is Fractal Soup

The reality of continuous data is a myth, because the idea of precise values is a myth. We sample or measure in finite elements, which have a finite size and thus an intrinsic uncertainty.

You might think that this sounds wrong and look for a way out: after all, mathematical curves have precise values, based on strict relations. If we plot a curve of system behavior, then we can sample with unlimited accuracy. This is true, but it is cheating. Naturally, one can compress data into a few parameters, if the whole story is known for all time, e.g., y=sin(x). Computers are not like this, because they are moved by unpredictable environmental influences. The number of parameters needed to describe their behavior is not known and may not even be constant. The values are hidden in "the environment," which is too big and complex to describe precisely.

Floating point representations can only calculate with limited accuracy; analog computers were only true representations of the calculations they were built to model, within certain tolerances. Can we ever achieve complete accuracy? The answer is no. No non-integer value can be measured with unlimited accuracy; thus no non-integer value can be fed into a computer with total accuracy; hence no exact

answer can be computed, even if the computer has infinite information resolution (which no computer has). In short, once we stray from the solid world of integers, everything is an approximation. This is the core realization of experimental science, and it is the fact which has implications for control of computers.

Digitization has consequences for *classification* and thus for system *state*. A digital representation is good enough to represent any state, but only within a certain tolerance. A digitization of fixed resolution implies an intrinsic uncertainty. Trying to control computers with simple push-button commands is like trying to counteract noise with single-tone filters. The amount of information is incommensurate.

In the mythological realm of truth and uncertainty, one might say that God is digital and the devil is continuous. Absolute truth and clarity can only be determined for discrete sets, at a fixed scale. Truth is not about infinite detail, it is about infinite clarity: or being able to see every digital grain. Truth is thus only obtainable in a low-resolution picture of the world. If one is forced to deal with continuously varying quantities with infinite detail, such as averages, then hellfire abounds in the uncertainties of never knowing how closely to look, never knowing what scale to choose. At some point one must give up and choose an arbitrary scheme of measurement (the measuring stick for comparisons). That is why anomaly detection and pattern recognition are hard. One cannot fail to have awesome respect for the brain as a processing device: it is fantastically successful at pattern recognition, better than any simple man-made algorithm.

There is one more twist which may turn out to be important in the quest to understand computer system behavior. Mathematically, a separation of scales is often possible only because the scales couple linearly. Nonlinearity is a concept which has been adopted from physics into common language, where it is loosely used to convey complexity. It occurs through dependency; a dependency is a strong coupling between parts of the system.

In a nonlinear environment, even small changes can resonate with large consequences. This idea was made popular by the notion of the butterfly effect, where the fantastical flapping of the wings of a butterfly in Japan could lead to equally legendary hurricanes in Florida (weather is nonlinear fluid dynamics). In other words, in nonlinear systems, it is difficult to predict the outcome of small perturbations. Nonlinear amplification tends to mix scales together, which is why nonlinear systems are difficult to understand (one often speaks of "chaos").

Digitization is clearly at the heart of computer behavior and measurement. We can be more precise about describing it and measuring its effects. The ability to discern state is clearly dependent on arbitrary choices. There is an inevitable nondeterminism (probability rather than certainty) in any system immersed in an environment. We need to recognize this and adopt strategies appropriately.

Computer Ecology Means Computer Immunology

Why is it not possible to control computers? The reason is simply that they are not predictable. We grow them in the laboratory, with simple push buttons, but we raise them in the wild, where environment pokes and prods them with far greater resolution. Unless we place systems in straightjackets, the influences of environment take their toll. Computer behavior is more intricate than any set of controlling commands we can practically build. This is an imbalance which provides the explanation for the tendency

In the mythological realm of truth and uncertainty, one might say that God is digital and the devil is continuous.

Competition is all that is left of strategy once digitization is suppressed.

of systems to disobey their controls. In order to compete with the complexity of the environment, one needs continuous measurement and regulative forces. *Competition is all that is left of strategy once digitization is suppressed.*

Machines get sick when this regulation fails. Sometimes regulation fails because a foreign element (virus) invades the system. Actually, it is not really important whether the element is foreign or not: the invading element might be "self" or "non-self." What is important is whether its influence is predictable in relation to the average state of the system. Will it lead to an instability? Will it drive the autonomous system in a new direction, different from the intended one? Sickness is not about right and wrong. Bacteria and viruses are not failures of a system, they are simply unusual input which alters its direction.

Inherent weakness is a precarious strategy for building technology, yet we build simple things for complex jobs. Fragility is no strategy for success. What is it that makes a system resilient and fit for its purpose? That it is under held rigid control, that it contributes to a larger whole, or perhaps that it endures under tough conditions? There are many answers. Much of the work in system administration has been about looking for simple answers to this complex problem, but we need to think of computers more as wildlife in a changing ecology. We need to embrace the intrinsic uncertainties faced when placing coarse digital animals into complex, intricate environments.

In the next part, we turn to entropy and what it means in the context of computer systems.

a logging and tracing facility for an embedded source code UNIX product

by Bob Gray

Bob Gray is co-founder of Boulder Labs, a digital video company. Designing architectures for performance has been his focus ever since he built an image processor system on UNIX in the late 1970s. He has a Ph.D. in computer science from the University of Colorado.

<bob@cs.colorado.edu>

I find myself designing yet another real-time, priority-based, multi-tasking system to use in a manufacturing plant. We are leveraging general purpose PC hardware and Source Code UNIX to control valves, relays, and motors and to monitor feedback with optical sensors and analog-to-digital converters. I've come to rely on a handful of tried-and-true methodologies for designing and building complicated systems.

This month I want to describe a debugging and tracing facility for multi-processing systems that I first used with PDP-11s in the late 1970s. The refined, current version, which runs under FreeBSD, is available from <*http://www.boulderlabs.com*>. The code should be trivial to port to any machine with a GNU C compiler and Sys V type shared memory. But regardless of your particular environment, the concepts are applicable to any real-time, multi-tasking system.

In an earlier article, I described the advantages of using a Source Code UNIX for embedded products. See my April 1999 article, (;login: vol. 24, No. 2) "Embedding Source Code UNIX in the Product" (<*http://www.boulderlabs.com/6.embedding*>). In this article, and successor articles, I'll share a number of design and coding methodologies that I have found successful.

A multi-tasking system often has complicated behavior. Various processes react to outside stimuli and internal interprocess communication. Tracing the system's activity is difficult when everything is working, and even harder during the development cycle when components aren't fully functional. The software described in this article is designed to track what is happening when the system is running, both during development and during product deployment.

The basic concept is that we wish to have control over the amount of logging data generated and present that information to the user in chronological order. During a debug cycle, we want to be able to increase the detail of logging in certain areas and decrease the detail in other areas.

The goals of the logging and tracing system are:

1. Low overhead so as to not skew what is being measured.
2. Easy visualization of what each process is doing and when.
3. A mechanism that not only allows the developers to control the amount of logging, but also to choose which messages get logged.
4. A mechanism that tracks the last *N* events without the need to flush data to disk.

Let's look at some sample output.

```
      A              B   C : D        E           F                  G
21:55:34.0258        C   177:1    chef_con.c    480:    This is the Chef version: 1.1
21:55:34.0272        C     0:2    chef_con.c    482:    chef_control running as CHEF
21:55:34.0278        F   177:1    frame.c      2537:    NOW START CHILDREN
21:55:34.0280        F   177:1    frame.c      1894:    before SendMsg: chData->numChildren=4
21:55:34.0285        M   177:1    motor.c       496:    Motor is alive waiting for event
```

Shared memory is the interprocess communication mechanism of choice when you're striving for minimal time impact of communicating

Column A is a high-resolution timestamp. You'll always see logging lines in chronological order. Column B is a symbol indicator of which process logged the message. Columns C and D are orthogonal parameters that control which messages appear in the log buffer. Details are given later. Columns E and F are the file name and source code line number. Column G is the string of the log message.

To satisfy goal number 1, we want a very low overhead logging/tracing mechanism. Ideally, there would be zero I/O and zero system calls. I'm willing to tolerate the cost of writing a message to memory, but a disk write is too expensive.

A naïve logging mechanism would have a buffer in each process of the multi-tasking system. This presents two problems. First, every process would have to provide an I/O mechanism to ultimately get the log information to the user. If a process were to crash, the logging information could easily be lost. Second, the relative sequencing of events in multiple processes is not easily visualized if every process logs its own data. (Granted, a post-processing step could merge the individual logs.)

Our mechanism designates one process as the parent that creates a piece of shared memory. All other processes "attach" to the shared memory. Any process can log information by acquiring write-access to the shared memory and copying its logging data. A circular buffer is implemented in the shared memory. One benefit of the circular buffer is that it keeps track of the last *N* events your system logged without any I/O. Like the "black box" in an aircraft accident, the last few things before the crash are usually enough. Optionally, you can request that the circular buffer be periodically flushed to disk or to a terminal. The code detects when the circular buffer wraps around and gives the user a clear indication that old data has been overwritten.

The logging software is leveraged from the BSD kernel circular buffer. If you look at /usr/include/sys/msgbuf.h on a FreeBSD system, you will see the code's heritage. I changed that base code to work in user-land instead of kernel space, and I added shared memory constructs to support the multiple process debugging. Shared memory is the interprocess communication mechanism of choice when you're striving for minimal time impact of communicating. I first used several of the techniques mentioned in this article when I programmed a message-passing system under DEC's RSX11M operating system in 1977. Several years later, shared memory made it into mainstream UNIX with Sys V (shmat, shmctl) and 4.2BSD (mmap).

Here is the essence of the logging:

```
circWrite(const char *p, int n) {

    lockResource(&mbp->msg_lock, &mbp->msg_busycnt);
    for(i=0; i<n; i++){
        mbp->msg_bufc[mbp->msg_bufx++] = *p++;
        ...
    }
    freeResource(&mbp->msg_lock, &mbp->msg_busycnt);
}
```

To enforce exclusive access, a process must first acquire the shared memory circular buffer by calling the lockResource function. Fortunately, most architectures (including the x86) have a "test and set" instruction that allows the lockResource function to be implemented without kernel assistance for an uncontested lock. My code uses the x86 cmpxchg instruction. Therefore, typically, only a few instructions are executed to gain access. (Thanks to Ron G. Minnich for posting his fastlock code many years ago on the Internet.)

Then, a tight loop copies a preformatted message into the circular buffer at memory writing speed. The freeResource routine is also fast. All of the code to log a message requires less than a microsecond on today's hardware. (One hundred instructions on a 100MIPS machine take 1 microsecond.) The chances of two processes needing to log at the same time are very low. For the contested access to the shared memory, we require the overhead of a system call which causes the process to context switch.

Over the years, I've learned that you want to build diagnostic tracing into your code from the start – it's harder to add it later. While a debugger, such as GDB, is invaluable in the development process, you also need logging and tracing in software products.

Over the years, I've learned that you want to build diagnostic tracing into your code from the start – it's harder to add it later.

Many years ago, I found Eric Allman's sendmail trace facility (see contrib/sendmail/src/trace.c). At runtime, it allows the user to turn on various debugging facilities at various priority levels. As a programmer, you sprinkle logging messages throughout your code, each message at a particular priority with a particular integer that Eric calls a "module ID." He chooses the following conventions for priorities:

```
#define FATAL 0
#define WARN 1
#define DIAG2 2
#define DIAG3 3
 ...
```

Module IDs are orthogonal to priorities. Each log message has both a priority and a module ID. At runtime, you can control which messages get logged by setting a priority level for each module ID. Once this vector has been initialized, at runtime, a message is logged if, for its module ID, the specified priority is lower than that of the corresponding vector entry.

It's time for an example. Assume that by a command line argument, I set the module ID vector as follows:

modID	Priority
0	2
1	2
2	2
3	2
4	2
5	3
6	1

And assume, sprinkled through my code, I have the following lines:

```
LOG(DIAG2, 3, 'M', "Got START from MOTOR");
LOG(DIAG3, 4, 'F', "Call to handle_TCP_read');
LOG(WARN,  5, 'G', "Command too long (%d)\n", length);
```

LOG is a macro with the following definition:

```
#define LOG(pri, mId, mod_name, fmt, args...)    \
{    \
if (mId>=0 && mId<TVLEN && tTdvect[mId] >= pri)        \
{    \
Log (pri, mod_name, mId, __FILE__, __LINE__, fmt , ## args);\}    \
}
```

You can see that we will "Log" the message "Got START from MOTOR" because tTdvect[mId] >= pri. Note also the third argument, 'M'. It's a symbol for the motor control

As products are deployed, we have found that these kinds of log files are often sufficient to diagnose a problem without any on-site visit.

process. "Call to handle_TCP_read" will not be logged because tTdvect[4] < DIAG3. "Command too long (%d)\n" from the 'G' process will be logged because tTdvect[5] >= WARN.

The Log function formats the message and has the basic form:

```
Log (int pri, int mod_name, int modId, char *file, int line, char *fmt, ...)
{
    gettimeofday(.....);     /* get system time stamp */

    n = snprintf (mbuf,LOGBUFSIZE,
       "%02d:%02d:%02d.%04ld %c%3d:%1d %-10.10s %4d: %s%s%s\n",
       t->tm_hour, t->tm_min, t->tm_sec, timeStamp.tv_usec / 100,
       mod_name, modId, pri, file, line, buf, errStr ? ": " : "",
       errStr ? errStr : "");
    circWrite(mbuf,n);
}
```

We made the decision to pay for one system call, gettimeofday, that gets an accurate time-stamp. The argument, module, is a letter that gives a symbolic identity to the process that logged to shared memory. For example, 'M' represents the motor control process. File and line are obtained from the C pre-processor using the __FILE__, __LINE__ directives. The variable number of arguments following the fmt string is handled with the va_list, va_start, vsnprintf mechanism.

The logging vector is initialized to priority 2 for all entries. That is, as code is executed, lines with a priority level of FATAL, WARN, and DIAG2 will cause logging activity. From the command line, we can override these defaults. For example, the parent process (the one that created the shared memory) could be started with the following debugging options which would be passed to the respective children processes, 'G' and 'M' (one is a GUI, the other is Motor control):

```
-dG3:25-33  -dM4:0-
```

Then the 'G' process would be told to adjust the vector entries 25–33 to priority level 3 and the 'M' process would be told to adjust its vector entries from 0 through the last, to priority 4. (Note, each process has its own tTdvect array.) Hence, logging would increase as a result of these runtime options. The syntax is:

```
debug_option   : '-d' [mod_name] pri ':' range
mod_name       : 'C' | 'D' | 'F' | 'G' | 'M'   (adjust for your system)
pri            : uchar
range          : uchar '-' uchar | uchar '-'
```

Finally, there are options to control the frequency of circular buffer flushing, if any, the file or file descriptor it goes to, the frequency of flushing, and the size of the circular buffer. Typically, it is the parent process that has this responsibility.

Over a couple of projects, we have found this logging and tracing facility invaluable during the development cycle. As products are deployed, we have found that these kinds of log files are often sufficient to diagnose a problem without any on-site visit. In the spirit of collaborative Internet development, we hope you find this software useful.

Thanks to Dave Clements and Tom Poindexter.

java performance

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>

The Cost of Exceptions

The Java language contains features for throwing and catching exceptions, where an exception is an abnormal condition such as an array index out of bounds or access through a null reference. Java exceptions are synchronous, that is, occur as the result of executing a particular instruction, and should be distinguished from UNIX-style signals delivered to a running process. A simple example of exception usage looks like this:

```
public class Simple {
    public static void main(String args[]) {
        int vec[] = new int[10];
        try {
            vec[15] = 37;
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.err.println(e);
        }
    }
}
```

In this program, an attempt is made to store a value at location 15 in a 10-long array. The attempt is made within a try block that specifies a catch clause for exceptions of type ArrayIndexOutOfBoundsException. If an exception is thrown, either explicitly via a throw statement, or implicitly by the run time system (as in this example), then the catch clauses are tried to see if a matching one can be found, and if so, control is transferred to the clause.

What is the cost of exception handling? Does it have a significant impact on the speed of your Java programs? In this column we'll look at a few examples of exception usage, and make some observations about the costs you should expect. As always, your results may vary from those described here, depending on the hardware you use and the particular Java Virtual Machine (JVM) you have. The JVM used for these examples is Javasoft's Hotspot 1.3.0-C version.

Overhead When Exceptions Are Not Used

How big a price do you pay for the exception feature when no exceptions are thrown? To find out, we'll use an example consisting of an inefficient sort algorithm, with a try block added to the inner loop:

```
public class Sort {
    static void sort(Object vec[]) {
        int len = vec.length;

        for (int i = 0; i < len - 1; i++) {
            for (int j = i + 1; j < len; j++) {

                // set up a try/catch block
                // to handle bad String casts

                try {
                    String si = (String)vec[i];
                    String sj = (String)vec[j];
                        int c = si.compareTo(sj);
                        if (c < 0) {
                            Object t = vec[i];
                            vec[i] = vec[j];
                            vec[j] = t;
```

```
                }
            }
            catch (ClassCastException e) {
                System.err.println(e);
                System.exit(1);
            }
        }
    }
}

public static void main(String args[]) {
    final int N = 10000;

    // create an array of object references

    Object vec[] = new Object[N];

    // populate array with String objects
    // of the form "abc1234"

    for (int i = 0; i < N; i++)
        vec[i] = "abc" + i;

    System.out.println(vec[0] + " " + vec[1] + " " + vec[2]);

    // sort

    long start = System.currentTimeMillis();
    sort(vec);
    long elapsed = System.currentTimeMillis() - start;
    System.out.println(elapsed);

    System.out.println(vec[0] + " " + vec[1] + " " + vec[2]);
}
}
```

In this example, an array of Object references is sorted, with an assumption made that the Object references actually refer to String objects. Before vec[i] and vec[j] are compared, they are cast from Object to String. It's possible that the assumption will be false, in which case a ClassCastException is thrown.

With the try block in place, the program uses 16218 units of time, and without the block, 16516 units, a negligible difference. So for this particular example and JVM, there's no cost associated with the try block.

Recycling Exception Objects

Let's get a little deeper into our investigation, and look at another example, one where an exception is repeatedly created and thrown:

```
public class Reuse {
    public static void main(String args[]) {
        final int N = 500000;

        // create an exception object

        Throwable exc = new Throwable();

        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++) {
            try {
```

```
                // either throw a new'ed object,
                // or a previously created object,
                // or a previous created object
                // with its stack trace filled in

                throw new Throwable();

                //throw exc;

                //exc.fillInStackTrace();
                //throw exc;
            }
            catch (Throwable e) {
            }
        }
        long elapsed = System.currentTimeMillis() - start;
        System.out.println(elapsed);
    }
}
```

This particular program requires about 3000 milliseconds to throw/catch 500,000 exceptions on a fast (800 MHz) machine.

We might wonder where the time is going. To find out, we can amend the example slightly. The first thing we do is repeatedly throw an existing exception object, instead of new-ing an object each time. When we do this, the time goes from 3000 down to 200 milliseconds. We might conclude from this that there's a lot of overhead in creating new objects.

This is so, but perhaps in a different way than we might think. When an exception object is created, an internal method fillInStackTrace() is called to record within the object details of the current state of stack frames for the current thread. We can try a third variation on the example to capture the time required for recording the stack trace. When an existing exception object is repeatedly thrown, but fillInStackTrace() is first called, the time goes from 200 to 2700 units. In other words, recycling exception objects does save you a lot of time, but at the expense of accurate stack trace information. The speedup may not be worth it.

Processing Many Catch Clauses

Another possible efficiency issue comes up when you have many catch clauses in a try block. The run time system must go through the clauses to find a matching one. An example of this situation looks like this:

```
class E1 extends Throwable {}
class E2 extends Throwable {}
class E3 extends Throwable {}
class E4 extends Throwable {}
class E5 extends Throwable {}
class E6 extends Throwable {}
class E7 extends Throwable {}
class E8 extends Throwable {}
class E9 extends Throwable {}
class E10 extends Throwable {}

public class Many {
    public static void main(String args[]) {
        final int N = 10000000;
```

```
        // create an exception of type E10

        //Throwable exc = new E1();
        Throwable exc = new E10();

        long start = System.currentTimeMillis();
        for (int i = 1; i <= N; i++) {

            // throw an exception and have it
            // caught by the E10 catch clause

            try {
                throw exc;
            }
            catch (E1 e1) {
            }
            catch (E2 e2) {
            }
            catch (E3 e3) {
            }
            catch (E4 e4) {
            }
            catch (E5 e5) {
            }
            catch (E6 e6) {
            }
            catch (E7 e7) {
            }
            catch (E8 e8) {
            }
            catch (E9 e9) {
            }
            catch (E10 e10) {
            }
            catch (Throwable e) {
            }
        }

        long elapsed = System.currentTimeMillis() - start;
        System.out.println(elapsed);
    }
}
```

If an exception of type E1 is thrown, it will match the first clause, while if an E10 exception is thrown, it will match a clause near the end of the list.

If the program throws an E10 exception, the time is 3578 units, while for an E1 exception, the time is 3407, a difference of 5%. So there's a difference here, but not a very large one.

Avoiding Exceptions

Sometimes there are cases where you can avoid exceptions altogether, and save time in doing so. Suppose you need to test whether an object is of String type. There are a couple of ways of doing so. One is to try to cast the object to a String, inside of a try block, and catch the ClassCastException that may be thrown. Another approach uses the "instanceof" operator instead of exceptions.

These two approaches can be programmed as follows:

```
public class Cast {

    // see if Object is a String by trying to cast it

    static boolean isString1(Object obj) {
        try {
            String s = (String)obj;
            return true;
        }
        catch (ClassCastException e) {
            return false;
        }
    }

    // see if Object is a String by use of the instanceof operator

    static boolean isString2(Object obj) {
        return obj instanceof String;
    }

    public static void main(String args[]) {
        final int SCALE = 100;
        final int N1 = 1000000;
        final int N2 = N1 * SCALE;
        boolean b;
        Object obj = new Object();

        // repeatedly call isString1()

        long start = System.currentTimeMillis();
        for (int i = 1; i <= N1; i++) {
            b = isString1(obj);
        }
        long elapsed = System.currentTimeMillis() - start;
        System.out.println(elapsed);

        // repeatedly call isString2()

        start = System.currentTimeMillis();
        for (int i = 1; i <= N2; i++) {
            b = isString2(obj);
        }
        elapsed = System.currentTimeMillis() - start;
        System.out.println(elapsed / SCALE);
    }
}
```

Some care needs to be taken in measuring the elapsed times, because the second approach using instanceof runs about 1000 times faster than the first approach that uses exceptions. In this particular example, there's no virtue in using exceptions, because the same problem can be solved in a simpler and faster way.

Summary

We've looked at several examples of performance issues with exception handling, and illustrated some techniques for optimizing performance. As always, it pays to be careful with performance tuning. It's usually best to write code in a natural, clear style, and not to worry too much about squeezing out the last little bit of performance. But when you're desperate to speed up a program, it's good to know where the time is going.

using new types and values in C9X

by Glen McCluskey

Glen McCluskey is a consultant with 15 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>

If you've followed the development of the C language over time, you're probably aware that there have been some additions to the original ANSI standard for the language. One of these updates was recently approved, and goes by names such as C99, ISO/IEC 9899, and C9X. In the next few columns we'll look at some of the features added to the language, and we'll use the term "C9X" to refer to the new features.

In this column we'll look at several new types added to C9X, and new ways of expressing values of particular types.

Bool

Bool is an integer type used to hold the values 0/1. It represents an obvious way to store false/true values. The keyword _Bool has been added to the language, so you say:

```
_Bool b = 1;
```

or:

```
#include <stdbool.h>

_Bool b = true;
```

Note that _Bool is not a macro defined in a header file but an actual keyword in C9X that is fully incorporated into the C type system. The macros false, true, and bool are defined in stdbool.h as 0, 1, and _Bool, respectively.

What happens when you try to use _Bool in expressions, like this?

```
#include <stdio.h>

int main()
{
    _Bool b = 1;
    b = b + 37;

    printf("%d\n", (int)b);

    return 0;
}
```

The Bool value is converted to an int, and then 37 is added to it. The result, 38, is nonzero, and so the value 1 is stored back into b. This same approach is used for conversions to Bool from other types, for example:

```
char* p;
_Bool b;

...

b = p;
```

This is equivalent to:

```
b = (p != 0);
```

and b gets a value of 0 if p is null, otherwise 1.

You can use Bool in bit fields, like this:

```
struct A {
    _Bool b : 1;
};
```

If you use Bool values in your program and you have, say, an array of values like this:

```
_Bool b[1000];
```

will the values each be stored using only one bit? The answer depends on the compiler you have. For example, a particular compiler might use a whole byte for each value.

Long Long

Long long is another new integer type, guaranteed to hold at least 64 bits. For example, you can say:

```
long long x = 123456789012345LL;
```

or:

```
unsigned long long x = 0xffffffffffffffffull;
```

You print long long values using the ll modifier of printf():

```
printf("%llu\n", 0xffffffffffffffffull);
```

The output is:

```
18446744073709551615
```

Complex and Imaginary

The keywords _Complex and _Imaginary are used in combination with floating types to specify complex data types. There are a total of six such types:

```
_Complex float
_Complex double
_Complex long double
_Imaginary float
_Imaginary double
_Imaginary long double
```

The header file <complex.h> specifies a macro I, that has the value of the imaginary unit, that is, the square root of –1. You initialize a complex value by saying:

```
_Complex float x = 37.0 + 47.0 * I;
```

The imaginary unit has the usual properties, for example:

```
I * I == -1
```

Here's a small program that initializes and multiplies two complex numbers:

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

int main()
{
      _Complex float c1 = 37.0 + 47.0 * I;
      _Complex float c2 = 57.0 + 67.0 * I;
      _Complex float c3 = c1 * c2;

      printf("%g %g\n", crealf(c3), cimagf(c3));

      return 0;
}
```

crealf() and cimagf() are functions used to obtain the real and imaginary parts of a complex number. When this program is run, the output is:

```
–1040 5158
```

In other words, when you multiply:

```
(37 + 47i) * (57 + 67i)
```

you have:

```
37 * 57 + 37 * 67i + 47i * 57 + 47i * 67i
```

or:

```
–1040 + 5158i
```

You can convert complex values to floating values and vice versa. For example, if you say:

```
_Complex double c = 77.0 + 87.0 * I;
double d;

d = c;
```

then d gets the value 77.0, and the imaginary part is discarded.

The C9X standard also specifies functions that operate on complex values, for example catanh() for complex arc hyperbolic tangent.

Hexadecimal Floating Constants

We've looked at several new data types that C9X provides. There are also new features for expressing values of specific types. One of these is the ability to express hex floating constants. For example, you can say:

```
float f = 0xf.fp+10f;
```

which has the value:

```
(15 + 15/16) * 2^10 = 16320
```

and:

```
double d = 0x8.0p-3;
```

with the value:

```
(8 + 0) * 2^-3 = 1
```

You print hexadecimal floats using the a specifier in printf():

```
printf("%a\n", 59.0);
```

The output here is:

```
0x1.d8p+5
```

Hex floating literals offer a natural way of expressing certain kinds of floating constants.

Compound Literals

Another new way you can specify values is through the use of compound literals. A compound literal looks like this:

```
(type){value1,value2,...}
```

Here are several examples of compound literals:

```
(int){37}

(int[]){1,2,3}

struct Point {int x, y;};
typedef struct Point Point;
...
(Point){100,200}

(_Complex long double){a + b * I}
```

The programming value of these literals is obvious; you don't have to explicitly name a temporary and initialize it. Here's an example that contrasts the old and new approaches:

```
#include <stdio.h>

struct Point {
      int x;
      int y;
};
      typedef struct Point Point;

      void f(Point p)
{
          printf("%d %d\n", p.x, p.y);
}

int main()
{
      // old way

          Point p = {100,200};
      f(p);

      // new way

          f((Point){100,200});

      return 0;
}
```

Compound literals are not constants and, if used within a function, can be initialized using non-constant expressions. And you can take the address of a literal, like this:

```
Point* ptr = &(Point){x + f(), y + g()};
```

When you use compound literals, it's important to note that each literal creates only a single object in a given scope. For example, in this code:

```
int i;

for (i = 1; i <= 10; i++) {
      Point* ptr = &(Point){i,i+10};
}
```

there's only one literal object, initialized at each loop iteration.

Compound literals give a hint of what some of the most important features in C9X are about – the ability to express a program more naturally. For example, it's quite possible to implement long long and complex data types using libraries, but it's more natural to include these features in the language itself.

the tclsh spot

by Clif Flynt

Clif Flynt has been a professional programmer for almost twenty years, and a Tcl advocate for the past four. He consults on Tcl/Tk and Internet applications.

<clif@cflynt.com>

The previous two "Tclsh Spot" articles described building a client-server package to monitor disk and network usage. Being able to monitor the current state of a server is useful, but it's even more useful if you can use that data to watch for a trend. That means saving and displaying some historical data. This article will describe using the BLT vector and barchart commands to show the historical data, and will describe the technique of using a canvas widget to attach a scrollbar to widgets (like the barchart) that don't normally support scrolling.

The Tk canvas widget is one of the workhorse widgets in the wish application. You can draw vector style images on a canvas, display images on it, and even display other Tk widgets on it. If you have an image larger than your display, you can link the canvas to a scrollbar to view a window into a larger area.

The syntax for creating a canvas widget is the same as for creating other Tk widgets:

Syntax: canvas *canvasName* *?options?*

canvas	Create a canvas widget.	
canvasName	The name for this canvas.	
?options?	Some of the options supported by the canvas widget are:	
	-background *color*	The color to use for the background of this image. The default color is light gray.
	-scrollregion *boundingBox*	Defines the size of a canvas widget. The bounding box is a list: left top right bottom that defines the total area of this canvas, which may be larger than the displayed area. These coordinates define the area of a canvas widget that can scroll into view when the canvas is attached to a scrollbar widget. This defaults to 0 0 width height, the size of the displayed canvas widget.
	-height *size*	The height of the displayed portion of the canvas. If *-scrollregion* is declared larger than this, and scrollbars are attached to this canvas, this defines the height of the window into a larger canvas. The *size* parameter may be in pixels, inches, millimeters, etc.
	-width *size*	The width of this canvas widget. Again, this may define the size of a window into a larger canvas.

Once you've created a canvas widget, you can create items on the canvas with the $canvasWidget create subcommand. The syntax for the create subcommand is:

```
canvasName create itemType coordinates ?-flag value?
```

The code to create a canvas with a box in the upper left corner would resemble:

```
set cvs [canvas .c]
grid $cvs -row 0 -column 0
$cvs create rectangle 0 0 20 20
```

Creating graphic objects and images on a canvas is quite simple. When you use the canvas widget as a holder for other windows, life becomes a little more complicated.

Tk defines each widget as being part of a hierarchy of windows. Each window (except the top level) is the child of some parent window. The windows are named using a period as the name separator. The main window in a wish session is named ".", just as in a file system the root directory is "/".

In a simple GUI application all the widgets can be children of the main window. This leads to names like .button1, .canvas, etc.

For more complex applications, you may need to use the frame or canvas widgets to group your widgets. This leads to widget names like .buttonFrame.quitButton or .canvas.barchart.

Most windowing systems propagate parameters (like window size) from a parent window to the child windows. While Tk will let you display any window within a canvas or frame, it's best to make the window you intend to display in another window a child of that parent.

This code would create a label as a child of a canvas, and display the label on the parent canvas, instead of on the main window:

```
set cvs [canvas .c]
grid $cvs -row 0 -column 0
set l [label $cvs.label -text "I'm on a canvas"]
$cvs create window 20 20 -window $l
```

One problem with any drawing package is the need to show a drawing that's larger than your display. The standard solution to this problem is to create a viewing window into the larger object, and move the viewing window around the larger window with one or two scrollbars.

The canvas widget can be treated as a window into a larger drawing area by setting the -scrollregion option to be larger than the canvas size. A scrollbar widget can be created with the scrollbar command.

Syntax: scrollbar *scrollbarName ?options?*

scrollbar	Create a scrollbar widget.	
scrollbarName	The name for this scrollbar.	
options	This widget supports several options. The required *-command* option is required.	
	-command "*procName ?args?*"	This defines the command to invoke when the state of the scrollbar changes. Arguments that define the changed state will be appended to the arguments defined in this option.
	-orient *direction*	Defines the orientation for the scrollbar. The direction may be horizontal or vertical. Defaults to vertical.
	-troughcolor *color*	Defines the color for the trough below the slider. Defaults to the default background color of the frames.

The wish interpreter handles the interaction between the canvas and scrollbar by registering a callback procedure with the scrollbar and canvas widgets. Whenever one of these widgets changes state, it will evaluate the registered script to update the other widget.

The code to create and display a canvas and scrollbar would resemble:

```
set cvs [canvas .c -xscrollcommand {.sb set} -height 20 -width 40]
set scroll [scrollbar .sb -command {.c xview} -orient horizontal}
.c configure -scrollregion {0 0 20 400}
```

```
grid $cvs -row 0 -column 1
grid $scroll -row 1 -column 1 -sticky ew
```

With the canvas and scrollbar widgets, we can enhance the network monitor.

The server part of this application was described in the previous "Tclsh Spot" (February 2001 *;login:*, p. 49). It sends unformatted data to the clients from whatever system utility is being watched.

The client is responsible for parsing and displaying the data. The goal of this design is to be able to use compute-intensive graphics and intelligence to monitor a system without adding extra overhead to an already overloaded server. A fallout of this design is that we don't need to modify the server in order to add more information to the client display.

The client uses the BLT extension's barchart command to build a barchart, and the vector command to hold the data being displayed.

The syntax for creating a barchart widget looks like this:

Syntax: barchart *name* *?option value?*

name	A name for this barchart widget. Using the standard Tcl window naming conventions.
?option value?	Option and Value pairs to fine-tune the appearance of the barchart. The available options include:
-background	The color for the barchart background.
-height	The height of the barchart widget.
-title	A title for this barchart.
-width	The width of the barchart widget.
-barwidth	The width of each bar on the barchart.

The barchart is created with code like this:

```
set Client(barChart) [::blt::barchart .bcht -width 600 -title\
    -title "Network Activity" -barmode aligned]
$Client(barChart) axis configure x -command getTicLabel
$Client(barChart) axis configure y -logscale 1
```

Once a barchart has been created, we can add elements to that barchart by creating a vector to hold the data with this code:

```
::blt::vector xvector(5)
::blt::vector yvector(5)
```

and then adding an element to the barchart with this command:

```
$Client(barChart) element create $name -label "$name" \
    -stipple [lindex $Client(stipple) $Client(count)] \
    -fg black -bg white \
    -xdata xvector -ydata yvector
```

As the client receives data, it parses the data, and sets the appropriate vector value, and the barchart automatically redraws itself to show bars of the appropriate height.

This is useful, but again, one goal for this client is to show historical data – like the data rates over the past few minutes.

If we were writing this type of an application in C, we might write something like this:

```
int dataValues[100];

...
addValue(int newValue) {
int i, j;
# Shift the values down, and add new value to the end of the array.

    for (i=0, j=1; i<99; i++, j++) {
        dataValues(i) = dataValues(j)
    }
    dataValues(99) = newValue
}
```

If we needed to, we could write similar code in Tcl to shift values in an associative array. However, if you are worried about performance, you should avoid large data-moving loops in interpreted languages.

One of the features of the BLT vector command is that you can delete an element at the beginning of the vector, and the rest of the values will shift to fill the empty space. You can add a new value to the end of a BLT vector with the special purpose ++end index. The Tcl code for sliding all the datavalues over and adding a new value resembles:

```
unset  dataValues(0)
set    dataValues(++end) $newValue
```

The previous example tracked the current values for number of bytes transmitted, received, and the number of collisions. To do this, we used a naming convention for the vectors we defined.

In order to get a list of the available vectors with the info globals command, we started each variable name with the characters DataVector, and then appended information about the type of data, and the device associated with this data. This leads to names like DataVector_x_eth0 and DataVector_y_eth0.

As a rule, it's better to organize data with an associative array, rather than a lot of individual variables. We can't quite do this when working with BLT vectors, since each vector must have a unique name.

What we can do is to use an associative array to hold the vector names. This means we don't need to worry about making sensible vector names. Instead, we can worry about making sensible array indices, which is actually an easier process.

When we create vectors, we need a unique name for each vector. The last version of this example used a clever naming convention involving the names of the devices and signals. This version uses a simple naming convention for the vectors (vector0, vector1, etc.) and lets the array index hold the information about the interface and type of data.

To generate the unique vector names, we need some way of creating unique numbers. The traditional technique is to use a variable and increment it after we create each vector. We can also use Tcl's ability to define procedures with a default argument and redefine procedures on the fly to create unique numbers when we need them.

This little gem was developed by Richard Suchenwirth. His original version of the code (along with many other clever little programming nuggets) is at *<http://mini.net/cgi-bin/wikit/526.html>*.

```
proc uniq {{val 0}} {
    incr val;
    proc uniq "{val $val}" [info body uniq]
    return $val
}
```

Using the associative array to hold the unique vector names, we can generate our list of vectors with code like this:

```
# Generate a set of unique vector names, and
# save those names in the DataVectors associative array

array set DataVectors [list \
    $name.y vector[uniq] \
    $name.y.rcv vector[uniq] \
    $name.y.xmt vector[uniq] ]

# Declare each of the new vectors to exist in global scope

foreach v [array names DataVectors $name*] {
    global $DataVectors($v)
}

# Create the new vectors

::blt::vector $DataVectors($name.y)(5)
::blt::vector $DataVectors($name.y.rcv)(200)
::blt::vector $DataVectors($name.y.xmt)(200)

# And initialize the history vectors

for {set i 0} {$i < 200} {incr i} {
    set $DataVectors($name.y.rcv)($i) 0
    set $DataVectors($name.y.xmt)($i) 0
}
```

The vector initialization lines may look strange if you know that Tcl does not support multidimensional arrays. The line set $DataVectors($name.y.rcv)($i) 0 looks like DataVectors is a two-dimensional array. What is actually happening is that the Tcl substitution phase is reading that line, and converting $DataVectors($name.y.rcv) to vector1, and actually invoking the set command as set vector1(0) 0; set vector1(1) 0, etc.

This set of code initializes a set of vectors for Y coordinates, but ignores the X coordinates for the bars.

By default, when two bars are defined to display at the same X location, the BLT widget will display the bars in the order they were defined, placing the latter bars on top of the earlier bars. This works when your data is scaled such that the latter sets of data always have smaller values than the earlier sets.

The barchart widget can also be configured to stack one set of data on top of the other, or to make the bars narrower and place them next to each other.

Using the -barmode aligned option allows us to create a single X vector for each barchart when the program starts, rather than creating a special X vector for each device we are watching.

Using these techniques, we can build the barchart for the current data with this code:

```
# Build a barchart for current data

set Client(barChart) [::blt::barchart .bcht -width 600 \
    "Network Activity" -barmode aligned]

::blt::vector dataXvector(5)
::blt::vector historyXvector(200)

for {set i 0} {$i < 5} {incr i} {
    set dataXvector($i) $i
}

for {set i 0} {$i < 200} {incr i} {
    set historyXvector($i) $i
}
```

Using five bars for the receive, transmit, receive errors, transmit errors and collisions is fine, but having just five sets of historical data is close to useless. We want to be able to scan several minutes of data, at least.

In order to display a few hundred bars, and make them more than a few pixels wide, we need a widget several thousand pixels wide. With the current monitor limit of 1600 pixels, this would limit us to 160 bars at 10 pixels wide. This is a bit under three minutes worth of history if we sample every two seconds.

While the BLT barchart widget cannot be directly linked to a scrollbar, it can be displayed within a canvas that has been attached to a scrollbar, as discussed at the beginning of this article.

This code will create a canvas 600 pixels wide, and pack an 8000-pixel-wide barchart into it.

```
# Build canvas, and then place the history barchart inside it

canvas .historyCvs -height 120 -width 600 -xscrollcommand {.histSB set}
grid .historyCvs -row 3 -column 0 -columnspan 4

scrollbar .histSB -orient horizontal -command {.historyCvs xview}
grid .histSB -row 4 -column 0 -columnspan 4 -sticky ew

set Client(historyChart) [::blt::barchart .historyCvs.bhist \
    -width 8000 \
    -height 150 \
    -barmode aligned \
    -title "" \
    -barmode aligned]
.historyCvs create window 1 1 -window .historyCvs.bhist -anchor nw
```

The BLT barchart widget supports drawing bars in various colors and stipples. For an application running on a color monitor, using colors to distinguish the bars is a good idea. Images in a magazine work better in black and white with stipple patterns. To display the transmit-and-receive data, I decided to invert the colors on a stipple pattern.

The Tcl associative array makes a handy lookup table to convert one color to the inverse. With this code, if the foreground color is white, then the inverse is $inverse(white) which is set to black.

This code creates two new barchart elements for a device, one for the transmit data and one for the receive data.

```
set inverse(black) white
set inverse(white) black
set color white

foreach dir {rcv xmt} {
    $Client(historyChart) element create ${dir}_$name -label "" \
        -stipple [lindex $Client(stipple) $Client(count)] \
        -fg $color -bg $inverse($color) -borderwidth 0 \
        -xdata historyXvector -ydata $DataVectors($name.y.$dir)
    set color $inverse($color)
}
```

These pieces of code, added to what was already in the client, will create the new bar-chart and new elements. The final piece is to put data into the history vector.

The Linux /proc/net/dev pseudo file reports data as running totals. What we want to display is the data for a single time period. We can handle this by saving the previous running total and subtract the current running total to get the per-interval value.

The associative array is a good way to save the previous values, and the Tcl info exists command will let us check to see if our program has a previous total to work with.

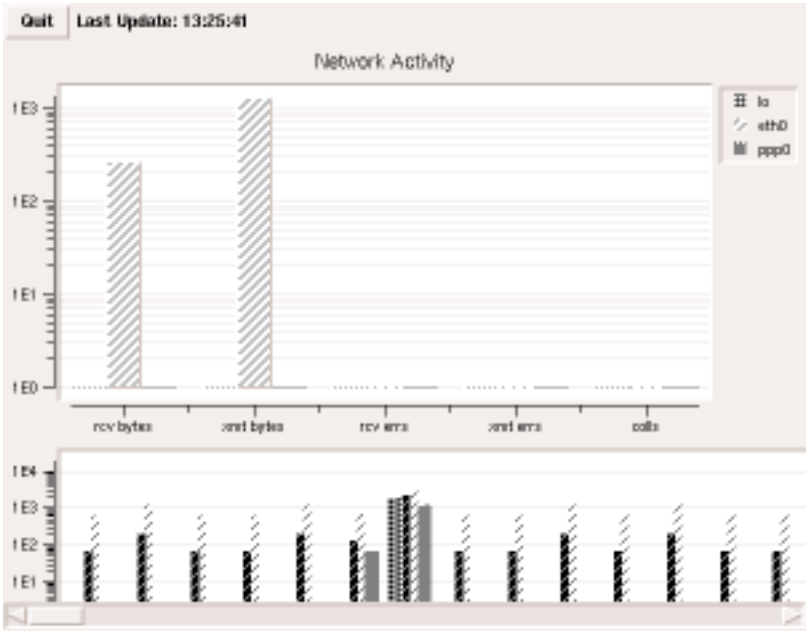
This code extracts the receive and transmit values from the raw data, converts the running total to the amount of data moved in the last interval and appends that value to the end of the appropriate history vector:

```
foreach direction {rcv xmt} \
            val [list [lindex $line 1] [lindex $line 9]] {
    if {[info exists Client($name.$direction)]} {
        set v2 [expr $val - $Client($name.$direction)]
        unset $DataVectors($name.y.$direction)(0)

        # Workaround to force vector to move values needed for some revisions
        #  of BLT
        set $DataVectors($name.y.$direction)(0)

        set
$DataVectors($name.y.$direction)(++end) $v2
    }
set Client($name.$direction) $val
}
```

When this is all done, the client generates images that look like this:



The code for this client and the code for the servers is available at <*http://www.noucorp.com*>.

introducing peep: the network auralizer

by Michael Gilfix

Michael Gilfix is completing his degrees in electrical engineering and computer science at Tufts University.

<mgilfix@eecs.tufts.edu>

Peep is a tool which provides an alternative to current network monitoring solutions. Peep creates an audio representation of network activity using natural sounds, providing large amounts of information in a compact, non-intrusive, and perhaps soothing form. Use of Peep to monitor your network is based on the concept of normalcy, where your network is functioning correctly if Peep "sounds right."

Why Peep?

Even though a good portion of our job as system administrators is to be as knowledgeable as possible about the state of our network at any given instant, it is difficult to use current approaches to live monitoring while completing other tasks. In general, one must intermittently suspend other work to check the monitor, which has profound negative consequences for accomplishing work efficiently, or perhaps wait (or pray?) for an email or page when something goes wrong.

These approaches are highly problem-centered and provide mainly negative reinforcement. Most tools are very limited in the domain they can address and report only when they discover a problem. In the meantime, the administrator is left in the dark, hoping the tool will do its job when the crucial time comes. These monitors do not regularly inform the administrator when the network is functioning well.

Peep's approach is to generate a realtime sonic representation of network state using natural sounds. Peep's audio output is meant to play in the background as a sort of white noise. With Peep, an administrator can keep tabs on how well the network is performing and what sort of activity is occurring at all times, without interrupting other work. In addition, Peep leaves all the subtleties of interpretation to the listener, thereby avoiding limitations on the kind of problem domain addressed.

Using audio to interface with the user brings several major benefits: it allows us to exploit our human instinct to notice deviations with little effort, to determine what "sounds right," and to discern singular important sounds from a collection of many sounds. These abilities are exercised continuously, with little or no conscious effort. Since computer interfaces mainly require the visual senses, using the audio senses to perform this unconscious processing does not interfere with our ability to do other work.

An audio interface also allows us to take advantage of our ability to do abstract processing. Instead of attempting the difficult and sensitive problem of determining when a network crisis has occurred or is about to occur, Peep provides contextual, continuous sound information and leaves interpretation to the listener. Decisions are then based not only on the quantitative measure of things, but also the relative amount and absence of things.

Yeah. But Won't That Get Annoying?

Nope. Most people envision audio tools as spewing some sequence of beeps or sounds that eventually annoy the listener and do more harm than good. Past approaches to audio monitoring have made use of beeps, midi, and singular sound samples to alert the user. These approaches, however, are greatly limited by what they can represent and still sound pleasing to the ear. For music to remain pleasant, one must limit one's represen-

tations to a limited number of relatively pleasing harmonic combinations. Singular and overly striking sounds are another dead-end, since we cannot tolerate these sounds in large measures (think ICQ!).

Natural sounds have an advantage over these approaches because they seem to "occur together" and "sound right" in virtually any combination. For example, birds, when singing in nature, have no coordination and yet they are still agreeable to listen to. Natural sounds also offer another advantage over conventional sounds – they have personality. We can map natural sounds to network events in the manner in which we think of them, making our representations more expressive.

Audio Representation

Since audio representation is such an important part of making an audio tool successful, Professor Alva Couch and I spent a substantial amount of time considering how to best represent network occurrences. Sound representation in Peep is divided into three basic categories: events in networks are things that occur once, naturally represented by a single peep or chirp; network states, or ongoing events, are represented by changing the type, volume, or stereo position of an ongoing background sound; while heartbeats represent the existence or frequency of occurrence of an ongoing network state by playing a sound at varying intervals, such as by changing the frequency of cricket chirps.

Peep represents discrete events by playing a single natural sound every time the event occurs, such as a bird chirp or a woodpecker's peck. These sounds are staccato in nature and easily distinguishable by the listener. We noted that certain events tend to occur together and found it convenient to assign them complementary sounds. While monitoring incoming and outgoing email on our network, we perceived that the two events were often grouped together, since both types of email were usually transferred in a single session between mail servers. To better represent this coupling between incoming and outgoing email events and make the representation sound more natural, we used the sounds of two conversing birds. Thus, a flood of incoming and outgoing email sounds like a sequence of call and response, making the sound "imagery" both more faithful to our network's behavior, as well as more pleasing to the ear.

State sounds correspond to measurements or weights describing the magnitude of something, such as the load average or the number of users on a given machine. Unlike events, which are only played when Peep is notified of them, Peep plays state information constantly and need only be signaled when state sounds should change. Peep represents a state with a continuous stream of background sounds, like a waterfall or wind. Each state is internally identified as a single number measurement, scaled to vary from extremely quiet to loud and obnoxious. The idea is that background sounds should be soothing while the network is functioning normally and annoying when an administrator should be inspired into action.

The idea is that background sounds should be soothing while the network is functioning normally and annoying when an administrator should be inspired into action.

Heartbeats are sounds that occur at constant intervals, analogous to crickets chirping at night. A common folk tale is that one can tell the temperature from the frequency of cricket chirps; likewise we can represent network load as a similar function. Intermittent chirps might mean low load, while a chorus might mean high load. Heartbeats can also report results of an intermittent check (or ping) to see if a given machine, device, or server is functioning properly.

Peep Architecture

Peep is based on a producer/consumer architecture where many client producers around the network gather information and send it to a few, centralized server consumers for playback. Configuration information is stored in a single configuration file

Perhaps the most difficult part of this process is choosing which sounds you prefer.

that is replicated for clients and servers around the network. Clients alert servers to network events via short UDP messages, keeping overhead to a minimum. This architecture allows for status reports from any number of network devices or nodes, as well as a great deal of flexibility when structuring a given network implementation.

To ease the management of Peep's distributed architecture, Peep uses a mechanism called auto-discovery and leasing. Upon startup, each client and server broadcasts its existence to the network once and the appropriate peers automatically "discover" each other. Because of the statelessness of UDP, a mechanism is required to ensure that clients do not waste network resources by sending packets to non-functional servers. This is accomplished through leasing. During their initial communication, the server and client exchange a lease time. Then, at intervals before the lease expires, the client checks in with the server and renews the lease. An expired lease indicates that a server is no longer functioning, and thus the client ceases sending its network information.

Peep's auto-discovery and leasing mechanism uses a domain-class concept to group clients and servers together. This class information is specified in the single configuration file shared by servers and clients. During the startup, each server or client broadcasts only to the subnets designated by its respective classes, and announces to those subnets which classes it belongs to. Following the initial broadcast, a list of hosts is maintained on both sides and all further communications are direct. Both clients and servers can belong to multiple classes at the same time, and clients can communicate with many servers concurrently.

Getting Peep Up and Running in Your Network

Integrating Peep with your network is a four-step process: download the source code and sound repository package from the Sourceforge site, build the server, configure your server and choose your sounds, and deploy the clients throughout your network.

The process of configuring servers and clients is relatively easy. An example configuration file comes with the Peep distribution that requires only a little modification to get Peep up and running within your network. Ample documentation is also provided with the Peep distribution in HTML format or can be obtained at <*http://peep.sourceforge.net/docs/peep-doc.html*>.

Perhaps the most difficult part of this process is choosing which sounds you prefer. KeyTest, a utility provided with Peep, is used for this exact purpose. KeyTest maps different keys to different events and states, and each keystroke plays the corresponding events or state on the server. This allows the user to experiment with changing stereo location, state volumes, and event priorities. My suggestion is to load all the sounds into the server and literally "play" the server to see how they all might sound together. KeyTest will support up to 24 different event sounds and 10 different state sounds at a given time. As this process can provide much amusement, it tends to be the lengthier part of Peep's setup time.

The last step is to deploy the clients provided with Peep. Currently, two clients are provided with the Peep distribution: Uptime and LogParser. Uptime reads state information from the UNIX utility "uptime," as its name would suggest, and reports a scaled measurement of the machine load and number of users to the server. LogParser, a real-time log parsing utility similar to Swatch, scans logs as data is appended and performs regular-expression pattern matching to extract event data. LogParser is a rather flexible tool, and the patterns it matches can be entirely customized in the Peep configuration file.

In addition, if the utilities provided with Peep do not meet your needs, all of the auto-discovery and leasing part of the Peep protocol has been encapsulated nicely within two Perl libraries provided with the Peep distribution. Example code exists in the documentation to help you write your own utilities quickly and efficiently.

Some Known Problems and Where We Are Going

One of the biggest problems with Peep is training your ear to recognize the intricacies of different network occurrences and make a complex diagnosis. This post appeared on slashdot shortly after the LISA 2000 conference:

> Since I'm on call, I'm looking forward to my first conversation with a monitoring guy after this is in place . . .
> MG: "Yeah, there's a problem with system XYZ . . ."
> Me: "How so?"
> MG: "Well, usually it goes 'ree-ree-tinktinktinktink,' you know? But right now it's going 'ree-ree-tinktink-bong-bong-tink'!"
> Me: "Is that 'bong' like a doorbell chime, or more like a big Chinese gong?"
> MG: "In between but more like a gong, I think."
> Me: "Well, shit."
> By nakaduct on slashdot – mike.muise@digital.com

But this example is part and parcel of dealing with inexperienced users of any system. Any system requires a user to become trained and "conditioned" to its proper use before attaining maximum benefit. In general, user feedback has been positive, and admins have reported that they have detected a large range of different behavior using Peep, most notably email spam. So, although training your ear may be of some concern, it has not been a problem for past users.

We are working on adding a recording feature to Peep that will save past events in a playback file for review. Currently, if you think you have heard some sort of anomaly, there is no way to go back and reevaluate the sound. A playback feature would allow admins to trade playback files amongst themselves to get second opinions.

We are also exploring visual playback methods that will provide visuals to network events similar to what a graphic equalizer does for sound. This will allow for a visual analysis similar to how Peep works, where a network is functioning correctly if the graphic just "looks right."

Summary

Whether we know it or not, we all have the ability to utilize our "peripheral" senses in doing our day-to-day work. Too many of us can instantly recognize the sound of a bad fan or a hard disk crash. We did not consciously study this or take a course in it. We learned it because these sounds form an integral part of our daily work environment. If we can add Peep to this environment, it is only a matter of time until we react to a cheerful chirp with the sure knowledge that our servers are working and that we can rest easy.

Obtaining Peep

Peep is freely available at <*http://www.sourceforge.net/projects/peep/*>. If you want to find out what this tool might sound like, there is an mp3 demo available on the home page under the introduction section.

"Well, usually it goes 'ree-ree-tinktinktinktink,' you know? But right now it's going 'ree-ree-tinktink-bong-bong-tink'!"

musings

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>

Tech stocks crashed, and I didn't leap from my office window. Not that it would have done any harm, since my office is on the ground floor. But it was amazing to watch as the paper value of my retirement fund plummeted. Alan Greenspan successfully punctured the tech stock bubble, while the President was talking the US into a recession so he can justify a tax cut (which will take effect long after any recession has ended). What a way to begin the millennium!

I was fortunate enough to be able to attend the LISA conference in New Orleans, which despite some rainy weather was still a respite from an unusually cold winter. Global warming, yeah sure! Please do not take me wrong, as I am thoroughly convinced that pumping carbon dioxide, particulates, and other compounds into the air for a hundred years has a measurable effect on climate. If you can see plumes of pollution from space (you can), it is easy to believe that the human race has had a measurable effect on climate. Today, we dump stuff into the atmosphere and hope that it will dissipate – just like people used to do with their sewage by dumping it into large bodies of water. Someday, our techniques for getting rid of gaseous waste will look just as absurd and primitive.

But I digress. The fourteenth LISA conference provided an embarrassment of riches – three tracks instead of two. In other words, there were two invited talks tracks competing with the paper presentation track, and it was devilishly hard to choose which session to attend. Given my personal focus on security, you probably can guess which sessions attracted me the most. Also, I know I can read the Proceedings, and unless I have a Peter Honeyman-like question to ask the poor paper presenter, I often decide to listen to an IT that will not appear in conference handouts.

If you missed LISA, get the Proceedings. I liked the FOKSTRAUT paper about extending Samba to handle "Windows machines determined to tell us their local passwords before attempting to give us the one we wanted." Although the solution, which involves caching those local passwords, sounds really scary, Beck and Holstead (University of Alberta) do recognize the problem and use a dedicated server, carefully secured, for this task.

The very next paper, "Designing a Data Center Instrumentation System," forced me to face a facet of security that I have blissfully ignored. For years I have been suggesting that people take advantage of all the floor space freed up in raised-floor areas that used to hold mainframes. By putting servers in secure areas, they can fix one of the biggest weaknesses in local security, physical access. After all, any UNIX or NT system administrator knows how to get access to any file on any server without knowing any passwords, right? Just reboot with the appropriate installation CD (or a floppy boot disk) inserted in the target system. Moving the systems to a secure area fixes the problem.

Er, except that it turns out that having people near the servers has been important as well. For example, I know what my servers sound like, so if a fan or hard drive bearing begins failing, the noise it makes is quite different than usual and I can do something about it before it becomes a catastrophic incident. Now, move those servers into a data center with servers behind glass rackmount doors, and "even the piercing sound of a piezo-electric alarm two rows away is drowned out" (to quote the paper).

The solution, developed by Bob Drzyzgula of the Federal Reserve Board, involves both "easy stuff and hard stuff." The easy stuff includes things you might already have thought of, like connecting all the serial console ports to a terminal server, and using

SSH for secure remote administration via the terminal server (which might be a Linux or BSD x86 system with multiport serial cards installed). The hard stuff was monitoring other problems, such as temperature, DC voltages, AC current, fan rotation, LED states, and control functions (such as a relay for reset or power on/off of the monitored devices). Drzyzgula chose to design his own boards based on an off-the-shelf microcontroller, and to use RS-485 (the basis for differential SCSI busses) for the physical communication layer. If you enjoy hardware-based approaches, and are not afraid of soldering irons, you should read this one.

The All-Electronic Home

The monitoring paper leads nicely into a really fun IT, given by Lorette Cheswick, assisted by her charming husband and one other family member. For those of us who have not visited the Cheswick home in lovely, suburban New Jersey, Lorette filled us in on just what you can do given an unused intercom system and a willingness to visit the local Radio Shack store and write some scripts. Ms. Cheswick described the amazing talking doorbell and using the intercom to deliver text-to-speech messages to her children, like "the school bus is leaving in five minutes."

I particularly liked the interface that takes Caller ID and turns it into the caller announced, another text-to-speech application. Bill Cheswick has written scripts that announce the closing values for the Dow as well as alerting the family to interesting astronomical events that can be seen from their yard (as well as when to go outside and where in the sky to look). When I suggested the "talking computer voice" to my wife, I got a big no. But then, she was the person who asked me (forcefully) to move the oscilloscope and the frequency generator out of the dining room, which was probably a good idea.

Still, having a voice announce telephone callers (instead of squinting at the Caller ID LCD) is appealing to me. So is being able to see who is at the front door and being notified that the garage door is (still) open, both things that the Cheswicks' system does. Having the NASDAQ closing value announced has been too depressing lately to think about, but I did start thinking about other things I would like to do. For example, I want to install CAT 5 cable when I have my house remodeled. Even if Drzyzgula does make good points about RS-485, CAT 5 does have certain advantages.

For instance, you can transmit power over CAT 5. I knew there were "unused" lines in the four pairs of twisted cables in CAT 5 and learned that people are now using these for sending power to devices that are not big consumers. My local Cisco rep sent me an announcement about "inline power over Category 5," as it is required for the Aeronet 350 wireless LAN. Personally, I am not interested in broadcasting over microwave bandwidths throughout my house, and will be happy to stick to wires. The Cheswicks largely use X10 controllers, which use the house's existing 110 volt circuits as a bus, and their intercom system, which my house just doesn't have.

The New Borg Look

Dkap, a USENIX member who still appears to be our only Borg, showed up in New Orleans with a new look. Instead of the Private Eye display, which dominates one eye and includes a rotating mirror for scanning, he had a Kopin display, a one-quarter VGA full-color display that takes up about a one centimeter square area in front of one eyeglass lens. The Kopin display is a great advance over the Private Eye; it's easy to view (he unclipped the display and shared it with many attendees), but the one-quarter VGA means xterm windows that can hold only 40 characters per line. And the Kopin costs as

When I mention wearables, many people respond "Yuck, who would consider wearing their computer?," and then their cell phone starts ringing.

much as a 17-inch LCD monitor (at a quarter the resolution). But prices should come down.

The rest of the rig now fits into a vest and includes two separate processors, the main server (which supports the Kopin display through an FPGA) and a disk server based on an IBM Microdrive. The two processor boardlets communicate using 100BaseT, and the power bus uses the power over CAT 5 I just mentioned. There is also an I^2C bus for peripherals, including wireless (802.11), cell phone, IR, and GPS. The entire rig, as worn by Dkap, weighs under two pounds, and he claims as much as 20 hours without recharging (try that with your laptop). The Web page for the new style wearable is *<http://www.media.mit.edu/wearables/mithril/>*.

When I mention wearables, many people respond "Yuck, who would consider wearing their computer?," and then their cell phone starts ringing. During the call, they pull out their Palm and consult their schedule, make a note, then put away the two computers they have just used. Yeah, who would want to wear a computer in public?

Dkap and the Mithril design still use the Twiddler, which I consider a terrible design for a one-handed keyboard. Chording keyboards should be designed for human fingers. Curl your fingers halfway, and place them on a desktop, and you'll notice they form an arc, not a straight line. The Twiddler is designed so that both right or left-handed people can use the same device, so it has straight lines of buttons like the left hand of an accordion. I hated that when I played the accordion and can't imagine picking it up again. There are other chording keyboard designs out there.

Security

Part of the LISA third track was devoted to security. I really enjoyed both Steve Romig's and Tom Perrine's practical advice about handling security incidents that involve the police. I had heard Ches' "Mapping Corporate Intranets" talk at a security conference, but recommend it as interesting if you have not already heard it.

And beyond LISA, things are still popping. January brought with it four new BIND security bugs, three providing buffer overflows. Someday, Rob Kolstad (if he is still running the LISA Game Show in the distant future) will be able to use a box under the Security column labeled BIND. The question will be, "What critical component of the Internet was also the most widely exploited server software in the year 2000?" Actually, this should read ". . . between 1998 and 2001?"

The announcement of more buffer overflows in BIND set up a storm of criticism about the Internet Security Consortium, the maintainers of BIND (isc.org>). In the end, Paul Vixie amply defended ISC, pointing out that any large body of code is bound to have flaws, and the BIND version 9 has been completely rewritten. Still, the important point I want to make is if you have ANY servers running older versions of BIND, replace them. You can convince BIND servers to cough up the version number in most cases by using:

```
dig @serverip version.bind. CHAOS TXT
```

where @serverip is the address of the DNS server you want to query. Note that script kiddies everywhere already know how to do this, or have scripts that do it for them, so you won't be the first to do this if you have any public DNS servers.

Summary

This issue of *;login:* contains summaries of LISA, so you should read them (and the Proceedings) if you want to learn more.

To summarize my own column, I'd like to remind you all that most people alive in the world today have, by definition, average intelligence. This is a no-brainer, right? Sure, until you try using some off-the-shelf software designed for those average users. About 18 months ago I wrote that the best user interface ever designed is the light switch: a simple-state machine with an obvious interface and a quick and appropriate response to the user's interaction.

While things will have changed by the time you read this, Northern California is still having rolling blackouts to deal with the decision to deregulate electricity in California and how the free market took advantage of this wonderful opportunity. The result, which has practically bankrupted billion dollar utilities like Pacific Gas and Electric, has led some people to suggest reviving nuclear power as a "clean option" to fossil fuels. Let's think about how we have dealt with the waste products of this clean option.

One by-product of nuclear reactors is plutonium, with a half-life of more than 20,000 years. Designing something that can safely contain plutonium for this time span has proven beyond our current capabilities. The best solution so far is to put the plutonium into nuclear warheads, which then must be carefully stored in highly secured areas because they are so dangerous.

Another option for dealing with so-called "depleted uranium" is to shoot it at your enemies as anti-tank ammunition. The US has managed to transfer over 30 tons of radioactive material to the Middle East this way. The people who came up with this solution to the nuclear waste problem should be given some kind of award.

Again, I digress. I consider it my responsibility, and hopefully yours as well, to act as intelligently as possible in an uncertain world. And don't forget to update BIND.

ISPadmin

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>

Remote Authentication Dial-In User Services

Introduction

In this installment of ISPadmin, I examine the lifeblood of any service provider's remote access system: Remote Authentication Dial-In User Services, or RADIUS. RADIUS provides the following functions to service providers in support of their dial-up subscribers:

- Authentication (who can and cannot have access to their network)
- Authorization (specify what services any given user can access)
- Accounting (track usage of services on their network)

In a nutshell, RADIUS is what makes an ISP's dial-up networks function sanely. There are alternatives to RADIUS (such as Cisco's TACACS), but they are not appropriate for anything but the smallest dial-up networks (10 modems or less), as they do not provide nearly enough functionality for service providers.

Exactly What Is RADIUS?

RADIUS is a UDP-based protocol developed by Livingston (now Lucent) expressly for their Portmaster Network Access Server (NAS) hardware in the early 1990s. The protocol is specified by a set of request for comments (RFCs), most currently RFC 2865 for Authentication and Authorization (commonly referred to as AA) and RFC 2866 for Accounting. Together the three functions of Authentication, Authorization, and Accounting are referred to as AAA.

The RADIUS protocol has seen many extensions over the years; a list of RADIUS-related RFCs in the references section. A number of draft Internet Engineering Task Force (IETF) standards relate to RADIUS, most prominently the replacement to the RADIUS protocol (aptly named DIAMETER). The references section also contains a link to the IETF RFC Web site as well as the draft IETF standards Web site.

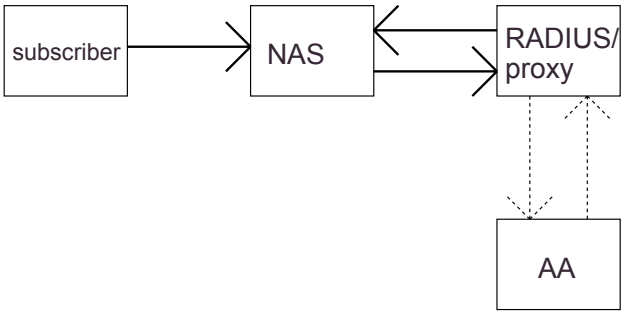


Figure 1

Figure 1 depicts RADIUS functions in the most generic way. A subscriber initiates a connection by dialing into a port on the NAS. After protocol negotiation with the subscriber's machine, the NAS sends a RADIUS "access request" to the RADIUS server to see if that subscriber is allowed onto the network. This request contains, among other things, the subscriber's username and password encrypted with an MD5 hash. (The RADIUS protocol also specifies optional proxy functionality, indicated by the dashed arrows to the box marked "AA" in the diagram.) The RADIUS server returns either a negative response ("access reject") in the case the user is not allowed or a positive response ("access accept") with the access rights for that particular session.

If the server allows access and if RADIUS accounting is configured on the NAS (which it should be for any commercial entity or organization interested in tracking subscriber usage), then the NAS will send an "accounting-start" record to the RADIUS server. Once the session is terminated, the NAS will send an "accounting-stop" record to the RADIUS server to account for the subscriber's usage for that particular session. Some NAS equipment will send what is known as "interim accounting" records

so that the RADIUS accounting server can track sessions in progress. Without such interim records, information about these sessions would be sent too late for use by certain types of applications which require it.

Small Provider Setup

A small provider's goals for AAA services are:

- Low cost, for both initial acquisition and ongoing maintenance
- Simple implementation

A small ISP's primary concern is cost, not features. As a result, a small ISP will probably use a free RADIUS server such as Livingston or Cistron. Also, they are not going to utilize RADIUS proxy functionality but, rather, have one or two RADIUS servers directly answering AA requests and logging accounting records. They will not likely be using the lightweight directory access protocol (LDAP) for end-user authentication or multiple servers to scale the load, as a small provider will not have the traffic to justify it.

Figure 2 outlines how a small ISP might set up their RADIUS infrastructure. Most NAS equipment is configured to be able to send RADIUS requests to (up to) two separate RADIUS servers, for ser-vicing both AA and accounting requests. This means that each NAS can specify up to four different IP addresses for RADIUS servers: a primary and secondary for AA, and a primary and secondary for accounting. (Figure 2 identifies the RADIUS servers by the labels "RAD1" and "RAD2".) Even the smallest ISP will likely utilize two RADIUS servers for redundancy purposes, preferably on separate subnets fed by separate switches/hubs and routers, if possible. This setup does require some additional work on the provisioning system to allow the account and password information to be sent to two RADIUS servers rather than simply one machine.

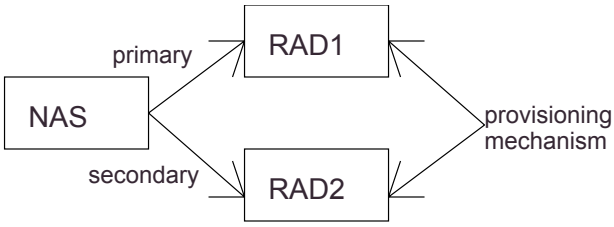


Figure 2

The RADIUS servers are usually set up to accept both AA and accounting requests. Although separate servers can be dedicated to AA and accounting, engineering each RADIUS server to accept all types of requests is a more flexible setup. It does, however, cause some additional work to reconstruct sessions on the back end, as the accounting-start record may go to one server and the accounting-stop record may go to another.

Several free or low-cost RADIUS servers are available to the small-scale operation, including the original Livingston server and its derivatives, such as Cistron and Freeradius.org servers. Also, Microsoft ships a RADIUS server with Microsoft Windows NT 4.0 Option Pack. The RADIUS software module itself is called "Internet Authentication Service," or IAS. These servers are covered in more detail in the "RADIUS Server Software" section below.

Medium/Large Provider Setup

The goals of a medium to large provider differ from those of a small ISP in the areas of functionality, extensibility, performance, and scalability.

A larger service provider will typically utilize a commercial RADIUS server such as Cisco's Access Registrar or create their own modified RADIUS server from one that has available source, such as the Livingston or Merit RADIUS servers. This modification is due to the fact that the original RADIUS servers with available source (Livingston and Merit) typically do not have the functionality and performance required for a 10,000 port or larger network. (According to the Merit Web site, the Merit RADIUS server was

licensed to Interlink Networks in June 2000; it is unclear if source is still available for the Merit RADIUS servers outside of Merit Network affiliates.)

A large provider is concerned about the performance and fault tolerance of the RADIUS server. They do not want a large customer's RADIUS server outage to affect the rest of their customers' ability to utilize their network. If not properly designed, one customer's outage can bring down even a 7,500-port network running Livingston or Merit. Also, wholesale customers usually have a number of authentication servers and would like more than one method of access to them: typically these modes include "round robin" and "fail over." In addition, the ability to set such parameters as server timeouts and number of retries is very desirable.

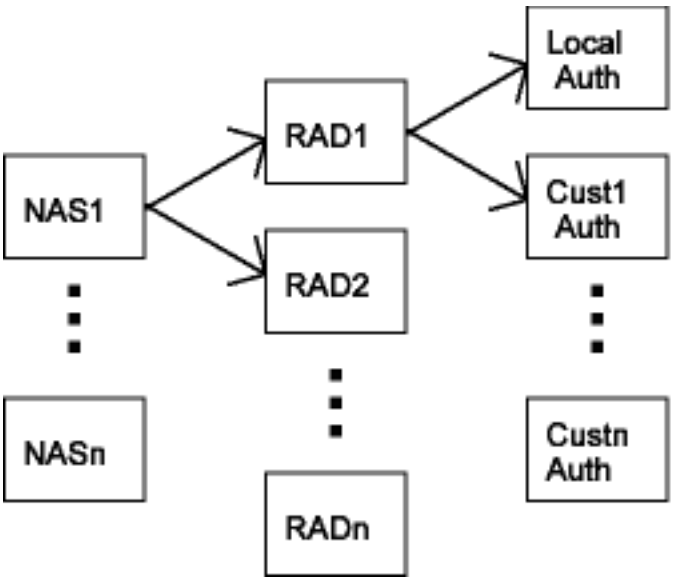


Figure 3

Figure 3 outlines a RADIUS implementation for a medium to large service provider. (Arrows are shown as one-way in the diagram for clarity.) The boxes marked RAD indicate RADIUS servers. These usually act as proxy RADIUS servers (as opposed to end authentication servers) in order to scale operations efficiently. Unlike a small ISP, a larger ISP will often wholesale their service to others, thereby utilizing the RADIUS proxy functionality. The diagram shows this by listing wholesale customer RADIUS servers below the "Local Auth" RADIUS server. "Local Auth" indicates a server that performs local authentication for the larger ISP. This would include retail accounts or virtual ISP services for customers who don't want to house their own servers in order to offer ISP type services (for example retailers, manufacturers, or affinity groups).

Authenticating End Subscribers

There are a number of methods to authenticate end subscribers. Most RADIUS servers support the following techniques:

- UNIX "passwd" file (or the NT equivalent in the case of MS IAS)
- RADIUS "users" text file (based on the original Livingston format)
- RADIUS "users" dbm file (hashed version of the plain text users file)
- SQL database
- LDAP

Only the smallest operations can utilize a UNIX "passwd" file. Most ISPs utilize one of the native RADIUS "users" file formats, usually growing from the plain text format to the hashed format as their business grows. A large service provider will utilize an SQL database or LDAP directory for authentication due to the scalability of these methods.

RADIUS and the Provisioning Process

When talking about RADIUS, one must always discuss provisioning. In a smaller ISP, provisioning is usually achieved by sending account information via a password file (or in the case of authentication via native Livingston RADIUS, a plain text users file or hashed users file) to the various servers that require it. Once the number of users gets too high (approximately 10,000), an alternative method must be used, as the performance of most commercial and open source RADIUS servers begins to suffer. After this threshold is reached, LDAP (or other directory service) or SQL is typically utilized.

LDAP is easily scalable, which is why it is recommended for large service providers. Once the maximum performance is reached on an LDAP server, another one is added

and linked into the LDAP tree. Another benefit to LDAP is the fact that the pluggable authentication module (PAM) directly supports LDAP, which makes integration into other applications (like email) seamless. SQL does not have the wide application support that LDAP has through the integration with PAM, which is why it is not utilized as often as LDAP.

LDAP Integration with RADIUS

Cistron and Livingston RADIUS include support of LDAP through PAM. However, this support is not nearly as thorough as a commercial product like Access Registrar. The Freeradius.org RADIUS server claims to have some built-in support for LDAP, but this author has no experience with it.

Cisco's Access Registrar 1.3 has a number of parameters which can be set when binding with an LDAP server (parameter on the left, example value on the right of the equal sign):

Name = ldap:cust.isp.net
Description = Cust
Protocol = ldap
IPAddress = 1.2.3.4
Port = 389
ReactivateTimerInterval = 300000
Timeout = 15
HostName = ldap.isp.net
BindName = uid=radius,ou=readers,o=isp.net,c=us
BindPassword = password
UseSSL = FALSE
SearchPath = ou=customers,ou=cust,ou=resellers,o=isp.net,c=us
Filter = (uid=%s)
UserPasswordAttribute = userpassword
LimitOutstandingRequests = FALSE
MaxOutstandingRequests = 0
MaxReferrals = 0
ReferralAttribute = <no value>
ReferralFilter = <no value>
PasswordEncryptionStyle = None
LDAPToRadiusMappings/
LDAPToEnvironmentMappings/

RADIUS Server Software

As with most software, RADIUS servers appear in two categories: open source and closed source (commercial). The first RADIUS server was Livingston's 1.x/2.x series of servers, which is the basis for the commonly used Cistron RADIUS server and others. The most recent version of the Livingston server is 2.1, which is available for free (without support) from the Lucent Web site.

The Cistron server is widely used. (It has several variants, including a MySQL back end; see the Cistron page for a complete list.) The Freeradius.org server is a follow-on to the Cistron server. The Freeradius.org server is currently in early alpha stage and not ready for production at this point. Development for these servers should merge at some future point. The current version of the Cistron RADIUS server is 1.6.4.

REFERENCES

Cisco TACACS+ starting point: <http://www.cisco.com/pcgi?bin/Support/PSP/psp_view.pl?p=Internetworking:Tacacs_plus>

Lucent INS (formerly Livingston Enterprises): <http://www.lucent.com/ins/>

RFC Web site: <http://www.ietf.org/rfc.html>

Draft IETF Web site: <http://search.ietf.org/search/brokers/internet-drafts/query.html>

RADIUS-specific draft IETF documents: <ftp://ftp.livingston.com/pub/archive/ietf-radius>

Livingston RADIUS servers (1.x and 2.1): <ftp://ftp.livingston.com/pub/le/radius>

Cistron RADIUS: <http://www.miquels.cistron.nl/radius/>

Freeradius.org: <http://www.freeradius.org>

Microsoft IAS: <http://www.microsoft.com/NTServer/commserv/deployment/moreinfo/ICS_FAQ.asp#4>

White paper on setting IAS: <http://zipdial.ziplink.net/docs/radius-nt.shtml>

Merit Networks: <http://www.merit.edu>

Interlink Networks (formerly Merit RADIUS): <http://www.interlinknetworks.com>

Cisco Access Registrar: <http://www.cisco.com/warp/public/779/servpro/operate/csm/nemnsw/car/prodlit/index.shtml>

LDAP man (articles on configuring LDAP): <http://www.ldapman.org>

Linux-PAM: <http://www.lyre-mit-edu.lkams.kernel.org/pub/linux/libs/pam/>

WideSpan from Bridgewater Systems: <http://www.bridgewatersystems.com/products/widespan/index.html>

RFCs related to the RADIUS protocol:

RFC 1227 SNMP MUX Protocol and MIB

RFC 2107 Ascend Tunnel Management Protocol – ATMP

RFC 2548 Microsoft Vendor-specific RADIUS Attributes

RFC 2607 Proxy Chaining and Policy Implementation in Roaming

RFC 2618 RADIUS Authentication Client MIB

RFC 2619 RADIUS Authentication Server MIB

[continued]

RFC 2620 RADIUS Accounting Client MIB

RFC 2621 RADIUS Accounting Server MIB

RFC 2809 Implementation of L2TP Compulsory Tunneling via RADIUS

RFC 2865 Remote Authentication Dial-In User Service (RADIUS)

RFC 2866 RADIUS Accounting

RFC 2867 RADIUS Accounting Modifications for Tunnel Protocol Support

RFC 2868 RADIUS Attributes for Tunnel Protocol Support

RFC 2869 RADIUS Extensions

RFC 2881 Network Access Server Requirements Next Generation (NASREQNG) NAS Model

RFC 2882 Network Access Servers Requirements: Extended RADIUS Practices

Another server which has been around for some time is the Merit AAA server (now licensed to Interlink Networks). Originally, the Merit server was distributed in two versions: AA and AAA. The AA version was free and the AAA was licensed for a fee. The future of the Merit AA server (the free version) for non-Merit-affiliated organizations is unclear. The AAA version will be maintained by the Interlink Networks organization. BSDi shipped a version of the Merit AA server as part of the distribution of BSD/OS.

Microsoft ships IAS (as part of the NT 4.0 Option Pack), which is a RADIUS server. It is an acceptable RADIUS server for smaller NT-only shops and the UNIX averse. The references contains a pointer to an excellent white paper covering the setup of a Microsoft IAS server.

A number of commercial RADIUS servers are available on the market. Two common stand-alone servers are Cisco's Access Registrar and Funk's Steel-Belted RADIUS. Many RADIUS servers are part of other larger software applications (e.g., ISP billing systems, provisioning systems, and policy management systems). However, stand-alone RADIUS servers are moving toward integrating policy management into them. WideSpan from Bridgewater Systems is an example of such a system.

Both the Access Registrar and Steel-Belted RADIUS/SPE (the Service Provider Edition) are designed expressly for the service provider market. Funk also has versions for the non-service provider market, as well as NT. Access Registrar was designed expressly for the telephone company market. Incidentally, Ziplink was the first ISP to deploy Access Registrar in a traditional ISP setting in October 1998.

Conclusion

RADIUS has three functions: authentication, authorization, and accounting. It is defined by a number of RFCs and is implemented by NAS equipment and software running on dedicated servers. Small ISPs design their infrastructure for low cost, while larger ISPs are more concerned about functionality, scalability, extensibility, and performance. Small ISPs tend to utilize open source RADIUS servers like Livingston 2.1 or Cistron. Larger providers tend to utilize commercial RADIUS servers like Access Registrar or Steel-Belted RADIUS/SPE and an LDAP back end.

Next time, I'll take a look at the topic of ISP billing systems and provisioning systems. In the meantime, please send your comments on UNIX, systems administration, the ISP industry, or related areas to me. I'd love to hear from you!

do you have both ORs in the water?

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.

<scj@transmeta.com>

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>

Our articles often focus on communication, and how to communicate more clearly. Most people believe they are communicating clearly most of the time, but others may not share that opinion! Sometimes the biggest problems are caused by the smallest words. We have previously discussed the use of How and Why, as well as Yes and No. This column will discuss OR, and next month we will talk about BUT.

When we say "A or B," we are asserting that at least one of A and B is true. If both may be true, the OR is called an "inclusive or" – if we are further asserting that A and B cannot both be true at the same time, the OR is called an "exclusive or." In the C language, we use '|' for inclusive OR, and '^' for exclusive OR.

In common usage, we use OR very sloppily:

"You had better clean up your room, or there'll be no supper!"

"Either we ship by Friday, or I'm out of a job."

"Joan and Jim or Bob will be coming too."

"Is this handled by Jack or Jill?"

The first two statements are using OR to express "if . . . then." If you do not clean up your room, I won't feed you. If we do not ship by Friday, I will be fired. In this sense, the OR does have a pretty clear meaning. Its function seems to be to deflect attention away from the speaker by making the statement appear to be some kind of a law of nature. So the second statement might more truthfully be stated as:

"If we don't ship by Friday, I'm afraid I will be fired."

It could also mean

"If we don't ship by Friday, my boss will be so angry he will fire me."

It is a good deal easier to respond to one of the two restatements than to the original. In the restatements, the problem is more clearly stated, and the speaker's concerns about it are explicit.

In general, restating an OR as an "if . . . then" will open up the discussion more. "How do you come to believe you will be fired?" "What else might we do to make your boss less angry?" "If we ship a partial order by Friday, and the customer is happy with that, how could your boss be upset?"

The third example of using OR points out that, unlike C, English is very sloppy about what OR actually means. The statement could be interpreted to mean that

Joan and Jim might come.

Joan and Bob might come.

Bob is coming alone.

All three might be coming.

The problem is twofold. English does not typically distinguish between inclusive and exclusive OR. And English doesn't have a precedence rule (like C does) that says that AND takes precedence over OR.

The most dangerous use of OR, however, is shown by the last example sentence. As we just pointed out, English doesn't typically distinguish between inclusive and exclusive ORs, so perhaps both should handle it. Just by making that statement, however, we limit our options severely. Perhaps it should be handled by Pat or Bill. There is a presupposition (we talked about presuppositions in an earlier article) that either Jack or Jill is the correct answer.

In a business setting, limiting our choices in this way is rarely the most productive way to think. Especially if we are having trouble making a decision, we need to look at the problem again and see if there aren't shades of gray between the two poles. Perhaps there is an "out of the box" solution that addresses a higher-level problem at the same time. Even the "inclusive" OR excludes many possible answers to our problem. By being more aware of our language, we can open the doors to alternatives that might be otherwise hidden.

We have seen that the short word OR has a lot of problems in regular English usage. Sometimes, it's a wimpy way of saying "if . . . then," limiting our thinking and turning attention away from the speaker. Sometimes its meaning is ambiguous. And sometimes it serves to artificially limit our alternatives. Becoming sensitive to these nuances can help you express yourself more clearly, and more easily find alternatives when responding to others.

you've been cracked . . . and now you're sued

by John Nicholson

John Nicholson is an attorney in the Technology Group of the firm of Shaw Pittman in Washington, D.C. He focuses on technology outsourcing, application development and system implementation, and other technology issues.

<John.Nicholson@ShawPittman.com>

The scene opens on a very serious man in a dark suit sitting on the edge of a desk in front of a wall of identical books. He says, "Has your privacy been invaded because a company exposed your personal information to unnecessary risk? Has a hacker stolen your identity or has your credit rating been damaged because a company should have done more to protect your credit card number? Did you lose your job or have you suffered embarrassment and humiliation because your private medical information was disclosed? You may have a claim. Call the lawyers of Able, Baker and Charlie. Your first consultation is free and, remember, there's no fee unless we recover for you. Able, Baker and Charlie – we're fighting for your privacy."

So far, that commercial hasn't been made. But given the increasing public interest (and paranoia) about information and data privacy, how long will it be before someone sues a company for damages because the company "allowed" that person's credit card number or other personal information to be stolen? It may not be long before ads like that are just as common as the other ads for lawyers that you see on late night TV. The purpose of this article is to help you understand the basics of this area of the law, with particular emphasis on the concept of negligence, so that you can work with your company's lawyers to develop a policy that minimizes the risk to your company of a lawsuit.[1]

Despite the apparent surge in seemingly silly lawsuits in the US, under US law every "tort"[2] claim must satisfy a four-part test in order for a plaintiff to succeed. Every tort claim must prove four basic elements:

1. Duty – the defendant must have a legal duty of care toward the plaintiff.
2. Breach of duty – the defendant must have violated a legal duty of care toward the plaintiff. Usually this violation is the result of "negligence" on the part of the defendant.
3. Damage – the plaintiff must have suffered harm.
4. "Proximate cause" – the defendant's breach of a legal duty must be related to the plaintiff's injury closely enough to be considered the cause or at least one of the primary causes of the harm.

Unless all of these are found to be true, the plaintiff in a lawsuit will not succeed.

When a Duty Can Exist

A duty can exist when there is a relationship between two or more parties. For example, a homeowner has a duty to protect guests from risks known to the homeowner but not to the guest. If a homeowner knew that a particular step on a staircase could not support weight, the homeowner would be liable if a guest were not aware of the risk and were injured by stepping on the broken step. According to the four-part test above, (1) the homeowner had a duty to the guest, (2) the homeowner failed to warn the guest about the step and breached the duty, (3) the guest was injured, and (4) the broken step was the "proximate cause" of the injury.

In a more technology-oriented situation, if a customer is providing information to your company as part of a transaction, usually such information is covered by your company's privacy policy. That privacy policy can create a duty and can bind your company

Even if there is no specific contract between your company and a person whose information gets disclosed because of something your company did or did not do, a court may still find that your company had a duty to take reasonable steps to protect that person's information.

to a level of behavior more stringent than that required by law. Even if there is no specific contract between your company and a person whose information gets disclosed because of something your company did or did not do, a court may still find that your company had a duty to take reasonable steps to protect that person's information.

What Exactly Is "Negligence"?

Negligence is defined as "failure to exercise the degree of care expected of a person of ordinary prudence in like circumstances in protecting others from a foreseeable and unreasonable risk of harm in a particular situation."[3] In the homeowner example, above, the homeowner may have been negligent in not warning the guest about the broken step. In the case of a person who claims that a company disclosed information in violation of that company's privacy policy, a court would determine whether or not that company complied with its own policy. In the case of breach of data security, a court would determine whether a company had been negligent by evaluating whether the company protected its information in a reasonable way given the cost of the protection, the sensitivity of the data, and what the company knew about the vulnerability that resulted in the information being disclosed.

Example Scenario

In the summer of 2000, a cracker used a password sniffer to compromise over 5,000 detailed medical records in an internal network at the University of Washington Medical Center (UWMC). The compromised information included the names, addresses, birth dates, social security numbers, and medical histories of over 4,000 cardiology patients, and additional information related to every discharged or transferred patient during a five-month period.[4]

Suppose, hypothetically, the cracker used that compromised information to steal the identity of one of those cardiology patients, and that patient decided to sue the UWMC for the damages, both financial and emotional, involved in repairing the patient's credit record. Despite the fact that someone else committed the identity theft, the plaintiff would be arguing that UWMC's failure to properly protect his medical records was the proximate cause of the financial and emotional harm suffered. In that situation, a judge or jury would look at whether, given the type of information being stored by UWMC, the UWMC was reasonable in the way it protected such information.

Going through the four-part analysis from above:

1. Did UWMC have a duty of care toward the plaintiff to protect the information provided by the plaintiff? Probably.
2. Did UWMC breach that duty?

In its analysis of breach of duty, a court would probably ask the following questions:

- What steps had UWMC taken to protect its information, and would a "reasonable person" have done things differently? According to *Information Security*, all information was taken over the Net and there were no firewalls in place.[5]
- What vulnerability was exploited to get privileges on the system?
- Had the vulnerability been made public and, if so, would a "reasonable person" have known about the vulnerability?
- Did a fix to the vulnerability exist and, if so, for how long prior to the breach was the fix available? Would a "reasonable person" have implemented the fix prior to the breach? For example, "More than 80 percent of successful attacks against NT-based Web servers exploited a vulnerability in RDS . . . [which] is installed by default on

NT-based IIS Web servers and is not commonly used by most Web sites. The RDS vulnerability is more than two years old and has had good patches available since July 1999."[6] If the UWMC vulnerability was something like this, that might weigh heavily against UWMC.

- Given the sensitivity of medical data and that the cracker used a password sniffer, would a "reasonable person" require users to use some kind of token in combination with a password? What were UWMC's requirements for password length and composition? How frequently were passwords required to be changed?
- Would a "reasonable person" have kept that type of data in that location? Was the database sitting on a Web server or was it somewhere else in the network? Given the power of a social security number for committing identity theft, was it reasonable for UWMC to track patients by social security number?

3. Was there damage? The plaintiff would have to show actual damages (which could include emotional damage).
4. If UWMC had a duty to the plaintiff and that duty was breached, and if the plaintiff suffered damage, was the breach by UWMC the proximate cause of the damage?

You might feel that the cracker in this hypothetical situation was the one who actually committed both crimes – first, cracking UWMC's network and stealing the information and, second, stealing the patient's identity, and that the only reason someone would sue UWMC would be to get money. However, the goal of this area of the law is to make people behave in a way that increases the relative safety of everyone. For example, if a store owner fails to properly lock his gun store, and a thief manages to break in and steal bullets which the thief then uses to shoot someone, then the store owner could be held liable. The reason for this is to encourage the store owner to recognize that his bullets create a hazard, and he should take appropriate care to prevent others from being harmed by that hazard. Tort litigation exists so that there is a cost-benefit analysis for people and companies to perform: the cost of preventive measures vs. the cost of the lawsuit.

Tort litigation exists so that there is a cost-benefit analysis for people and companies to perform: the cost of preventive measures vs. the cost of the lawsuit.

So What Do I Do?

Your first step should be to develop and implement, as part of your overall security policy, a procedure that tracks security risks (both external and internal) as they are identified, evaluates their potential risk to your business, identifies the appropriate fix, schedules a date for the implementation of the fix, and includes a follow-up procedure to ensure that the fix was properly implemented. For example, your policy should include:

1. Regular reviews of the relevant security vulnerability sources (i.e., Bugtraq, NTBugtraq, security reports published by software vendors, virus reports, security researchers, the various cracker Web sites, etc.) and, if appropriate, a procedure to ensure that such reviews are performed

In a diverse environment, your company may have multiple people responsible for various platforms and/or software packages, or your company may have various administrators with responsibility divided by geography. It's important to make it clear who will have the ultimate responsibility for monitoring security issues related to each platform or software package.

2. A determination of how the identified vulnerability applies to some aspect of your business

NOTES

1. This article provides general information and represents the author's views. It does not constitute legal advice and should not be used or taken as legal advice relating to any specific situation.

2. A "tort" is some damage, injury, or wrongful act done willfully or negligently for which a civil suit can be brought.

3. Merriam-Webster's *Dictionary of Law*, 1996, as published on <*http://www.findlaw.com*> as of Feb. 6, 2001.

4. *Information Security*, vol. 4, no. 1, January 2001, p. 24.

5. Ibid.

6. Peter Tippett, "Sweat the Easy Stuff," *Information Security*, vol. 4, no. 1, January 2001, pp. 30–31.

For example, a security hole that lets a script kiddie put graffiti all over your Web page can be embarrassing to your company or might result in your taking down your page until you can plug the hole. If your Web page is just information about your company, this might not be a big problem. If your Web page is the means by which your customers order, that's a different matter. It's important to understand how the vulnerability could impact your business if it were exploited.

3. A rating of the risk represented by the security issue (i.e., critical, high, medium, or low) based on the potential impact of the security issue to the business (in terms of lost business, public perception, potential cost, etc.)

4. A schedule for the implementation of the relevant fix for the risk (i.e., all critical fixes will be implemented within one day, all highs within one week, etc.)

5. A follow-up procedure that checks whether fixes were actually installed and, depending on the importance of the security issue, verifies whether the fix actually solves the problem.

A follow-up procedure could vary depending on the rating of the issue. For example, you might want to ensure that all fixes for critical issues are implemented, and use statistical sampling for the rest. Alternatively, you might want to ensure that all fixes for a mission critical system are performed, regardless of rating.

Once you've completed your procedures and made them part of your routine, do an audit of how your system stacks up against the known threats. This might involve having a "white hat" security attempt to penetrate your network. Use this as an opportunity to test your priority ratings as well. If a problem someone rated as "low" allows the penetration team to take control of your system, then you might need to reevaluate that rating.

Conclusion

It may not be long before people begin suing companies for information disclosures that result from a company's network being cracked. Companies need to develop the policies and procedures that will protect both the information (which is the primary goal) and the company in case the company is ever sued in relation to such a disclosure. Will having a security policy like this in place keep you from getting sued? No, although it might make it less likely. A security policy that includes the procedures described above won't prevent you from being attacked, and it won't prevent you from being sued because of an information disclosure. It will enable you to prioritize and understand the known risks to your system, and it will put your company in a better position if you ever are sued, including potentially protecting your company from punitive damages.

mobile hoarding and dynamic grouping

by Ahmed Amer

Ahmed Amer is a PhD candidate at UC Santa Cruz. As a member of the Computer Systems Laboratory he is involved in systems research with a particular focus on data management for mobile systems and alternative storage architectures.

<amer4@cse.ucsc.edu>

The USENIX Scholars Program *<http://www.usenix.org/students/scholar.html>* provides support for student stipends, tuition, and other expenses for students with exceptional research ability and promise. This article is an example of the kind of work resulting from this program.

Motivation

Mobile computing is growing more practical with ever-decreasing hardware sizes and power requirements. We also see a growing dependence on larger and larger repositories of data, and with limited bandwidth and the possibility of frequent disconnection, it is still not a simple process for a user to "take their computer and go."

With improving wireless communication technology this problem remains far from being solved. Wireless bandwidth does not approach the bandwidth available with conventional wired networks and is more susceptible to unforeseen service disruption (e.g., driving through a tunnel, signal interference).

One promising approach to allow for more effective mobile computing is file hoarding, the caching of a user's data to their local system's hard drive, with the goal of allowing disconnected operation or simply better performance over slow and unreliable network links.

Despite significant developments in file hoarding algorithms, none are available in a form that allows widespread general use. Whether the actual hoarding process requires extensive user input, or whether parameter selection requires extensive experimental tuning, we feel that a major obstacle to general availability of file hoarding as a useful tool is an insufficient level of automation. Our work proposes a model for file clustering and grouping based on statistical predictors (relationship estimators) that is amenable to automatic parameter tuning.

Description

Our system is based on a two-phase process: interfile relationship estimation, and the grouping of files to produce minimal intergroup relationships (see Figures 1a and 1b).

Relationship estimation uses previous file access events to provide a statistical model of file access relationships. This is similar to the task performed by predictive file caching. This includes graph-based systems that use access windows, or more recent work utilizing context modeling techniques, which can also be seen as providing probability estimates for subsequent file access events.

In Figure 1 (see next page), a relationship estimator in our system is required to accept a stream of file access events and produce a weighted graph G(V, E, W) of file relationship estimates. Here V is the set of files, E is the set of significant edges, and W is the set of associated weights (relationship estimates). Relationship estimates are weightings that are proportional to the probability of a particular file being accessed after another file. It should be noted that the resultant graph will be of a fixed maximum degree, since it is infeasible to track relationship estimates for "all other files," which would require quadratic space in the number of files in the file system. Fixing the degree of the graph is equivalent to placing a limit on the length of the adjacency list for a particular file. This restriction reduces any algorithm linear in the number of relationships to being linear within the number of files. This allows us to produce a feasible solution for the grouping

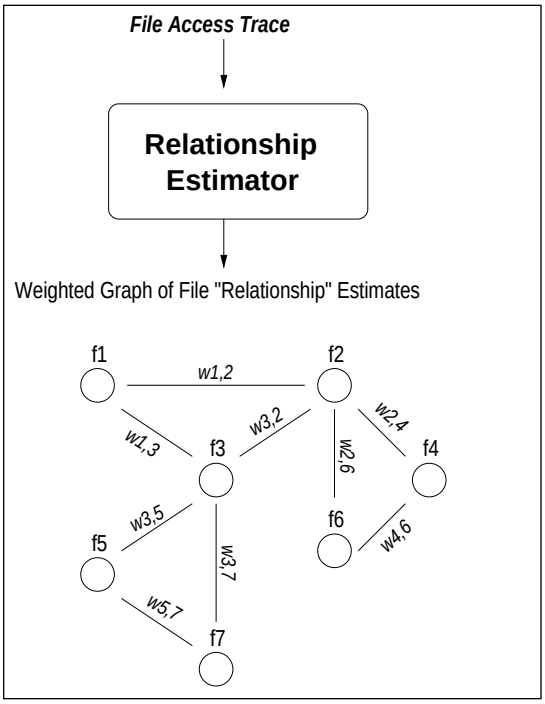


Figure 1a – Relationship Estimation

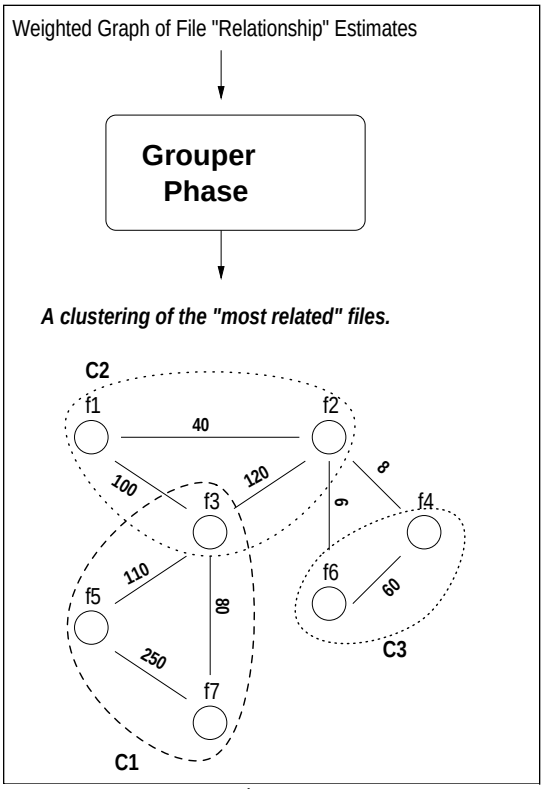


Figure 1b – Grouping

Given a stream of file access events, relationship estimation builds a graph (of strictly limited degree) estimating the level of interfile relationships, which is used by the grouping phase to divide the graph into minimally related groups

problem and also limits state-space requirements to a constant factor of existing file system metadata.

It is important to note that there are multiple mechanisms for measuring this "likelihood." We have considered several classes of such predictors and have produced a novel predictor, Noah, that identifies strong pairings among files with minimal state space and computational requirements.

In the second phase, grouping, we are given the graph of interfile relationship estimates, and we attempt to divide the graph into minimally related groups/clusters. Intuitively, we wish to group together the files most likely to be accessed within a short period of each other.

After dividing the weighted relationship graph into clusters of related files, the final step in the process of selecting files for hoarding is ranking the clusters. This is done by assigning a score to each cluster and is necessary when a single highly related cluster cannot fill the mobile store. In summary, we employ a ranking mechanism for the generated clusters, and use this ranking to select the clusters to be imported to the local system for hoarding purposes. There are several ranking metrics that can be used, varying in complexity from simple LRU up to more elaborate algorithms like file aging.

A final note on our approach involves the automatic tuning of operating parameters. We continuously attempt to avoid any manual adjustment of operating parameters for our algorithms. This is greatly facilitated by the strong machine-learning community within the University of California, Santa Cruz's School of Engineering.

Further Applications

We evaluate groupings generated by our approach against file access traces to verify their usefulness for mobile file hoarding. And yet, if we relax the requirement for cluster ranking, we can directly use the clusters we have generated in the grouping phase for purposes of data placement on storage media elements. In this scenario, the limit on cluster size would be directly proportional to the capacity of a single media "element." Media elements can be any units of a storage medium that invoke a high-latency operation for a switch between them. The limit on the total size of all combined clusters is analogous to the limit on total available storage space. (A subtle difference between placement and hoarding restrictions is that in hoarding, each file size is counted only once, whereas for placement, duplicate copies will use double the physical storage space, as opposed to simply state space for hoarding.)

Figures 2a and 2b. show the effect of increasing the latency/bandwidth gap on system throughput. For the tertiary store we assume typical access characteristics of a tape or M.O. disk media changer, while for the secondary store, throughput is increased to model a scenario where raw bandwidth can be increased to almost 1GBps – which is feasible given enough parallelism.

Examples include existing automated tertiary libraries that incur a heavy performance penalty for media replacement. A more general example is the trend in storage technology toward higher and higher bandwidth, with limited improvement in access speed. This is true of magnetic disks, optical disks, and even communication networks.

Placing data into clusters that have a minimal likelihood of intercluster transitions can ameliorate a wide range of growing systems problems. Constructing such clus-

ters is exactly the purpose of our two phases of relationship estimation and grouping. We focus on data hoarding applications which, thanks to the highly automated nature of our statistical predictors, promises to be performed with minimal user intervention.

Status

Work on relationship estimation is largely complete, and our most recent development, the Noah algorithm for low-cost online evaluation of file pairings, has generated two papers currently under review for publication. Noah maintains accurate pairings of files while tracking only two candidate successors for each file, and adapts to variations in access patterns on a per-file basis. Our most recent results for grouping, and optimality measures for online relationship detection, are currently being prepared for possible submission at the next USENIX annual technical conference in Boston.

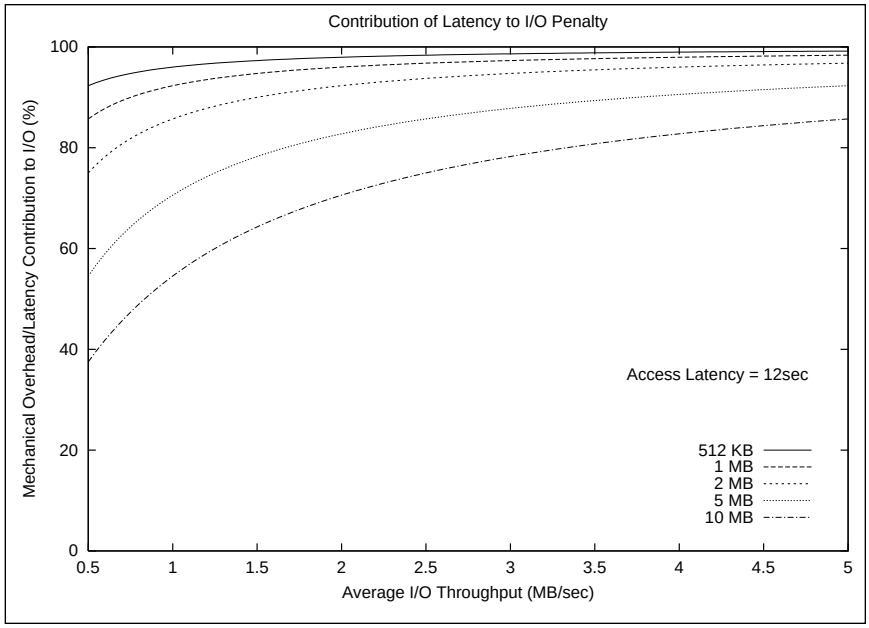


Figure 2a – Tertiary Store

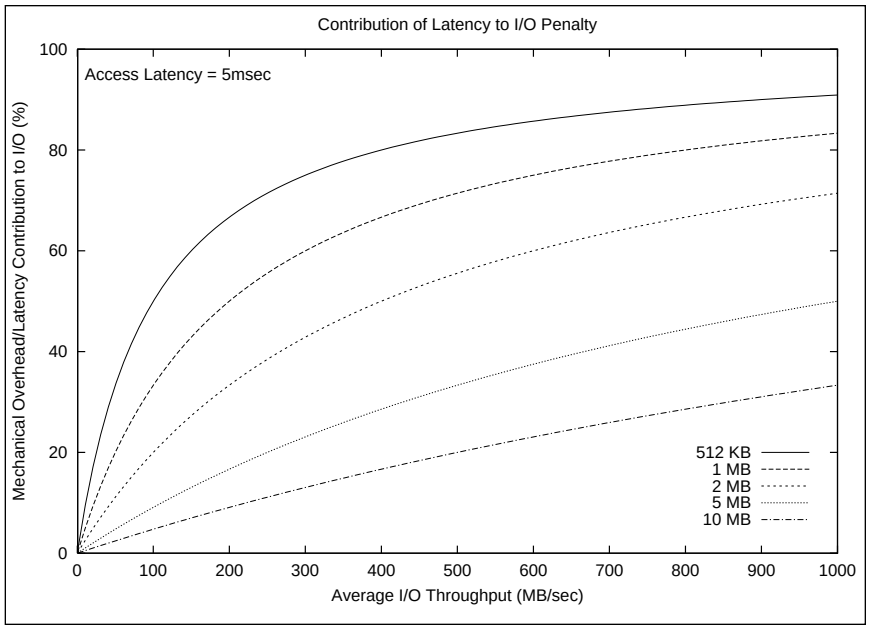


Figure 2b – Secondary Store

Acknowledgments

We are grateful to all the members of the Computer Systems Laboratory for their continuous feedback, support, and valuable discussions. We are also grateful to all those who provided us with detailed storage system traces, including G. Kuenning, D. Roselli, and especially J. Wilkes and the Hewlett-Packard Company, and R. Golding of Panasas. Our most extensive multi-year traces were made available by M. Satyanaryanan of Carnegie Mellon University, through the greatly appreciated efforts of T. Kroeger in processing and conversion.

This project is being conducted under the supervision and guidance of Professor Darrell D.E. Long, in the Computer Science Department of the University of California, Santa Cruz. This is one of several systems projects under investigation in the Computer Systems Laboratory, which has recently undergone significant expansion.

Further Information

Links to Web pages relating to this research can be found at
<http://www.cse.ucsc.edu/~amer4/research/>.

Further links to the Computer Systems Laboratory at the University of California, Santa Cruz, including summaries for research projects, members, and affiliates can be found at: *<http://csl.cse.ucsc.edu/>*.

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>

BOOKS REVIEWED IN THIS COLUMN

AD HOC NETWORKING
CHARLES E. PERKINS, ED.
Boston: Addison-Wesley, 2001. Pp. 370. ISBN 0-201-30976-9.

BLUETOOTH REVEALED
BRENT A. MILLER & CHATSCHIK BISDIKIAN
Upper Saddle River, NJ: Prentice Hall, 2001. Pp. 303. ISBN 0-13-090294-2.

THE FREE BSD CORPORATE NETWORKER'S GUIDE
TED MITTELSTAEDT
Boston: Addison-Wesley, 2001. Pp. 401+CD-ROM. ISBN 0-201-70481-1.

NETWORK PROGRAMMING WITH PERL
LINCOLN STEIN
Boston: Addison-Wesley, 2001. Pp. 754. ISBN 0-201-61571-1.

SOLARIS INTERNALS
JIM MAURO & RICHARD MCDOUGALL
Upper Saddle River, NJ: Prentice Hall, 2001. Pp. 657. ISBN 0-13-022496-0.

THE BLENDER BOOK
CARSTEN WARTMANN
San Francisco: No Starch Press, 2000. Pp. 350+CD-ROM. ISBN 1-886411-44-1.

RHCE LINUX
KARA J. PRITCHARD
2nd Ed. Scottsdale, AZ: Coriolis, 2000. Pp. 386. ISBN 1-57610-828-7.

There really are a lot of goodies this month. In fact, there are so many that I may end up being briefer than usual. We appealed for volunteer reviewers in the last issue. Come on! Tell me your interest(s) and I'll try you out.

Networking

I've now been using the ARPANET/Bitnet/Internet for about 25 years. I'm continually amazed by the ubiquity of both addresses and URLs. The growth from the four sites of December 1969 to the well over 100 million of December 2000 is breathtaking. So is my expectation that the Net is truly pervasive.

Elsewhere in this issue you'll see my thoughts on COOTS 6. But it's notable that one of the things central to Stu Feldman's keynote is mobility, what I used to call "location-resistant computing."

Charles Perkins has put together a really good anthology of important papers on mobility. *Ad Hoc Networking* is bound to become vital as we move toward the IP-on-everything world envisaged by Vint Cerf years ago and now coming into reality. While the military uses (Freebersyser and Leiner, pp. 29–51) are the most obvious, as each of us travels more and wants to access information wherever we are, downloading a road map or a theater schedule, finding out a sports statistic or a Supreme Court dictum, the formation of transient networks becomes vital.

I don't have the time or space to itemize the contents, but I was particularly taken by Perkins and Bhagwat (pp. 53–74) on DSDV; Johnson, Maltz, and Brock (pp. 139–172) on Dynamic Source Routing; and Toh and Vassilion on "The Effects of Beaconing on the Battery Life of Ad Hoc Mobile Computers" (pp. 299-321).

This is a very impressive set of papers.

In his keynote, Feldman mentioned Bluetooth. In his introduction, Perkins does, too. So it was with pleasure that I turned to Miller and Bisdikian.

Bluetooth Revealed is one of under a half dozen books on Ericsson's wireless specification. It's a relatively easy read and contains a useful bibliography.

Some of the largest networks in the world (e.g., Yahoo! and Hotmail) run on FreeBSD. Mittelstaedt has made the job of replicating those users a lot easier. The step-by-step is well done; the CD contains FreeBSD 4.2, XFree86 3.3.6, and hundreds of third-party packages.

Stein has modeled his networking book on Rich Stevens' exposition of TCP/IP. It's another major piece of work. If you're a Perl programmer involved with networking you will need to have this. I don't know where Stein gets the time or energy to produce good books as he does.

Internals

I enjoy reading internals books. Bach and the 4.3 and 4.4 volumes occupy a prominent place on my shelves. So does Vahalia. Now, along comes a volume by Mauro and McDougall that can be put with the other four.

The volume takes us through the kernel, but I thought the treatment of the algorithms and data structures was really fine. There's a useful bibliography and the index is good enough to enable you to employ the book as a reference.

book reviews

3-D Graphics

It took several thousand years for artists to discover ways to represent perspective and distance on a two-dimensional wall, canvas, board, or writing surface. Over the past 30 years, ways of representing space on a flat screen have increasingly occupied our time. (How I recall those SIGGRAPHs with hundreds of screens showing ray-tracing!)

About two years ago, the first version of Blender was released on the Internet by Tom Roosendael. It has been revised and improved since then, as have so many other open-source tools. Wartmann has now provided us with the first English introduction. Blender is useful for both the Web and for video, and Wartmann has written several excellent tutorials. The CD-ROM contains Blender 1.8 for Linux, FreeBSD, IRIX, Solaris, BeOS, and something called Windows.

Resurrections

By and large, I don't review second editions. This month I'm making an exception.

Kara Pritchard's Linux book is far more than an exam cram course: it is really a superb introduction to RedHat Linux. It has been thoroughly updated, and better still, it's shorter than the first edition.

Book Reviewers Needed

;login: (more specifically, Peter Salus) is looking for reviewers for books that deserve greater coverage than can be afforded within the Bookworm format. If you are interested, contact <*peter@matrix.net*>; feel free to suggest what topics you are interested in, and which book or books you might like to review.

DESIGNING STORAGE AREA NETWORKS

Tom Clark

Boston: Addison-Wesley, 1999. Pp. 202.
ISBN 0-201-61584-3 (paperback).

Reviewed by Steve Reames

Anyone who has been watching the information technology industry has noticed the increasing levels of excitement about Storage Area Networks or SANs. Marketing hype aside, SANs can provide improved reliability, ease of scalability, and simplified management of storage resources in large IT operations.

SANs have grown out of the open systems client/server world where servers running different operating systems communicate with multiple clients. LAN technologies – such as the ubiquitous Ethernet – were developed to enable individual computers to communicate with any of the servers. Most servers have their own private bank of disk drives and one or more tape drives to support the mass storage needs of their users. SANs place a network between the server and the storage, allowing multiple servers to access the same bank of disk or tape drives. Fibre Channel networks, with a data rate of 100MBps, are the current interconnect technology of choice.

Although Fibre Channel has been a standard since 1995, both Fibre Channel and SANs have remained in relative obscurity until the last two years. With the proliferation of SANs in enterprise network facilities, there is a need for books that explain the intricacies of SANs and Fibre Channel networking. Tom Clark's new book is a limited introduction to that world.

This small book, 200 pages long and 1/2-inch thick, is clearly intended to be an introductory text. The first three chapters are actually quite good and provide straightforward explanations of what can be a complex technology. Chapter 4 starts well but quickly becomes bogged down in the details of loop address negotiation in Fibre Channel Arbitrated Loops. Although this information is good for the technologist, it is a bit too much for a first exposure.

Subsequent chapters change to an erudite style of writing more suited to college textbooks or IEEE transactions. The information is there, but well-hidden in its abstraction. Paragraphs after paragraphs seem to go by without anything being said. Chapter 5, "Fibre Channel Products," is an encyclopedic-like descriptive alphabetical listing of hubs, switches, fabric modules, and other devices. Each product is described adequately, but the relationship between them remains unclear, almost as if each section was written by a different person.

The diagrams are sparse and poorly done. Standard clip-art symbols are used throughout, though this is not a weakness in itself. The problem is that the same graphic is often used to represent a hub, switch, fabric module, or Ethernet hub, and the item is often unlabeled. Diagrams that appear later in the book intermix all these items with Fibre Channel and Ethernet cables, but the connections remain undifferentiated. Using a different line weight, dashes, or even shades of gray would help immensely in figuring out which lines are Fibre Channel and which are Ethernet (a pretty important difference, in my opinion). One diagram in Chapter 5 purports to illustrate Raid mode 1+0 when in actuality it is only showing Raid mode 1.

Unfortunately, there are not many books on the market for Fibre Channel or SANs. If you have need for a brief introduction to SANs, then this book alone can provide a quick overview of the terrain, though I would stop reading at the end of Chapter 3. But if Fibre Channel and SANs are becoming an important part of your operation, I would suggest looking at some of the other more recent publications.

book reviews

PC HARDWARE IN A NUTSHELL

ROBERT BRUCE THOMPSON AND BARBARA FRITCHMAN THOMPSON

Sebastopol, CA.: O'Reilly and Associates, 2000. Pp. 500. ISBN 1-56592-599-8.

REVIEWED BY RIK FARROW

I have sat in on PC-building BoFs at USENIX conferences and listened to some of my friends state they positively will not build or upgrade their own PCs. I actually built my first computer by soldering connectors to the motherboard and other S-100 bus card slots, a computer that not only worked, but one that I was able to sell several years later after adding a hard disk to it. So the idea of assembling a PC from parts really appealed to me.

The reality is that the naysayers are *almost* correct. Building your PC is difficult, because not only are the possibilities endless, so are the configuration problems. This is where *PC Hardware in a Nutshell* comes in.

The Thompsons have their own Web site, which includes updates to the material in the book: <*http://www.hardwareguys.com/*>. One of the things I noticed in the book was most of the emphasis is on building PCs for various flavors of Windows, so if you want help building a system that will work specifically with Linux or a BSD-variant, you are better off visiting VA Linux or BSDi, as both sites sell systems customized to support these UNIX versions.

Where I needed the most help understanding PC hardware, this book does a good job. And that is in the area of memory, motherboards, and processors. For example, you can only use certain processors with certain motherboards for a couple of reasons: the physical connections between the processor and the motherboard (Slot 1, Slot A, Socket 1, etc.) and the various support chips for that processor, such as the frontside bus, have to match. I didn't realize that I couldn't just stick in a faster Pentium III processor, reboot, and have it run at a higher clock rate. Nope, you have to jigger your motherboard with numbers and/or BIOS settings to provide the correct clock rate (usually some fraction of the processor speed, such as one-third) before your new CPU will clock faster.

I would have loved to see some illustrations of the various slots and sockets – this book has no pictures. But, then again, that is simply my own desire to see what they are talking about. In reality, you don't look at motherboards to decide which ones to buy – you buy them based on their specs, chipsets, processor support, slots for memory and cards, and most importantly, their reputations for running the operating system of your choice.

I really enjoy one of the authors' major points: it is better to spend more money on a faster disk controller or more memory than on a faster processor. Spending $600 instead of $60 for your processor will only double your performance – not a lot of bang for your buck when you are only planning on spending $1,500.

I recommend *PC Hardware in a Nutshell* with the caveat that it will take more than what you can get out of this book to learn how to build your own PCs. It is a useful reference, not a tutorial on what is a difficult subject. Or, to quote from the authors' preface, "This should really be called *PC Hardware in a Coconut Shell.*" A useful tome.

standards reports

Some Standardization News

by Keld Simonsen

Keld Simonsen is active in ISO standardization, particularly in internationalization, POSIX, C, C++, and making it all available on the Web.

<keld@dkuug.dk>

Here are some updates on work in ISO standardization, in the field of IT, internationalization, and character sets.

JTC 1 – the group responsible for standardization of all IT in ISO and IEC – had a meeting in Tromsø, Norway, in November. Tromsø is a little below 70° North, and the days were shortening considerably during our stay – two weeks later it would have been always pitch dark.

There will be a trial to have a number of ISO standards available for a modest fee, say $25, for binary copies. Another trial will be continued for another free set of ISO standards; today about 50 standards are available this way. See *<http://isotc.iso.ch/livelink/livelink/fetch/2489/Ittf_Home/PubliclyAvailableStandards.htm>* for this list.

A proposal that participating companies could buy a voting membership for $25,000 a year on a level with national bodies was amended to a trial in which everybody in participating groups could take part on an equal level with national experts. Personally, I think it would have been very problematical if a dozen big firms had been allowed to buy a majority for an ISO standard, and I welcome everybody's individual participation in the trial without a fee. However, you still need to pay for your own travel.

A proposal to always use MS Word 7 as the internal document format in JTC 1 was changed to allow other document formats, including PDF, .txt, HTML, MS Word 6, WordPerfect 5.1, and RTF. I am happy that JTC 1 did not choose to only allow a non-standard document format, which is not fully supported with macros, for example, on some platforms, including Linux.

This is my pet area and there have been quite a few developments in standardization since my last snitch report. In August the convener of WG20 – the internationalization working group in ISO – reported in a personal note to the parent group SC22 that he thought that WG20 should be disbanded, a few projects transferred to some other standardization groups, and the rest of the projects cancelled. One of the main points was that the technology being standardized was arcane, as it builds on the C/POSIX locale internationalization model. Another point was that this area best be left to the industry to standardize. One project – the i18n API project ISO 15435 – had not progressed to the first ballot after three years and should, the convener thought, therefore be cancelled.

SC22 had quite a discussion on this and by a small majority decided to continue the project for one year and, furthermore, to ask WG20 whether the group itself thought it should be disbanded (the discussion report was a personal report from the convener and had not been discussed officially in WG20 before submission). In the WG20 meeting in November, a majority in the group disagreed with the view of the convener. WG20 also decided to send the cultural registry standard (which registers POSIX locales, charmaps, etc.) for its first ballot, to send the enhanced POSIX locale standard TR 14652 for its final DTR ballot, and to begin an addendum for the new sorting standard ISO 14651 covering the additions of characters to the ISO 10646 UCS standard. I think that this latter discussion derives from the USA's L2 group, which is very oriented toward character-set issues and which also represents the USA in the ISO character-set group SC22. The L2 group also has close con-

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

nections with the Unicode Technical Committee, as they hold all their meetings together. Some of the WG20 group work is in direct competition with the Unicode work. Other US groups are quite supportive of the WG20 efforts: the C and C++ groups have implemented the WG20 guidelines on what characters/letters can be used in extended identifiers, while the new revision of the COBOL standard will use the specification in the enhanced locale standard on how to map from uppercase to lowercase for all of 10646. I hope that the countries can come to an agreement on common standards about internationalization, and that individual countries will not try to either push their own standards through or sabotage international standardization in this area.

ISO 10646 – the huge character standard – is now being extended beyond the 16 bits. This is ISO 10646-2, which is now out for its final FDIS ballot. It contains mainly an extension with many ideographic (Chinese/Japanese/Korean) characters, and a few exotic scripts, plus some characters for language declarations. Two bytes are thus not enough any longer for UCS. The UTF-8 format of UCS is now increasingly being implemented in UNIX and Linux. Kde is doing all its messages in UTF-8, and glibc 2.2 supports conversion of all messages to UTF-8 via the gettext package and iconv. Glibc 2.2 even supports TR 14642-style transliteration if the execution character set does not contain specific characters in a message. The ISO 10646-1:2000 standard is now available for about $50 on a CD from ISO. This is not that expensive compared to the paper version that is priced according to normal (expensive) ISO rates. A number of 8-bit character sets are being finalized, including a character set supporting Romanian, and a revised 693 standard that covers most Latin letters with accents.

SAGE news

SAGE Elections

SAGE ELECTS NEW EXECUTIVE COMMITTEE FOR 2001–2003 TERM

The SAGE Executive Committee positions for the 2001–2003 term are as follows:

Strata Chalup
Barbara Dijker
Tim Gassaway
Geoff Halprin
Trey Harris
David Parter
Peg Schafer

Not elected:

Bryan C. Andregg
John Sellens
Andres Silva

For the first time, voting for the SAGE Executive Committee was conducted electronically.

Total number of SAGE members eligible to vote: 4861
Total number of votes cast: 606
Number of postal ballots: 3
Response rate: 12.5%
Total invalid ballots: 0

The newly elected SAGE Executive Committee members met in Berkeley, California, on March 9–10, 2001. For more information about the SAGE Executive Committee, please see:
<*http://www.usenix.org/sage/people/Current-Board.html*>.

SAGE Certification Project Update

by Lois Bennett
Member, SAGE Certification Committee
<lois@deas.harvard.edu>

The SAGE Certification Project continues to move forward. Patrons continue to be approached, the job description for Director of Certification has been approved by the Policy Committee, and the process of selecting marketing firms to help promote the Project is underway. Meanwhile, test writing for the first level examination has begun in earnest. However, the process of writing the actual questions has prompted a need for clarification about the basic aims of the first level of certification.

When certification was first embarked on, it was the intention of the SAGE Executive Committee that it be a vendor-neutral, platform-neutral, certification test concentrating on underlying principles and accepted practices. Eliminating such biases is essential, but hardly straightforward in practice. The questions are being written from the Knowledge, Skills, and Abilities (KSA) used in the occupational analysis of 1999, which sound UNIX-centric because nearly all the survey respondents were UNIX System Administrators. And as we had anticipated, writing questions that are both specific and vendor/platform–neutral is not easy.

But the Certification Policy Committee (CPC) believes that UNIX or NT bias can be removed. These issues absorbed much of the discussion at the February meeting, and some important clarifications were made. The result was to refine the focus of the SAGE Certification Project, as follows:

> The Certified System Administrator (CSA) is a single, vendor neutral, platform neutral test concentrating on the underlying principles, concepts, and accepted practices within each KSA area. All questions developed by the Test Development Committee (TDC) are to be derived from the KSA document already supplied. This will be augmented at a later time with further documents. All questions are to be platform/vendor independent, although it is recognized that draft questions may include platform specifics to be later discussed, and revised so as to be platform-independent.

This focus should in turn make for clearer guidelines for the test writers. It was acknowledged that the existing KSA base needs to be reviewed and potentially supplemented with additional accepted practice areas.

The point, in other words, is to write questions that probe the understanding of underlying computing principles, rather than the peculiarities of NT, UNIX, or other operating systems. This reflects our "market differentiator": the value of this test, as opposed to others on the market, is that it will establish accepted practice standards at a level beyond specific operating systems.

The test writing that has been done so far has been informed by a workshop on good test writing practice led by Gordon Waugh of The Human Resources Research Organization (HumRRO) at the TDC meeting, after December's LISA conference. The two-day workshop gave basic principles of writing effective multiple choice tests and an opportunity to come up with examples and critique them for practice.

Fifteen SAGE level IV system administrators are on the team, including four women. Two members of the CPC are serving as test developers to serve as liaisons and to further the project.

The TDC will be meeting again in April for an intense session to review the questions that have been written. Each writer is tasked with drafting 40 questions, with a goal of 450 total items for the beta testing in June.

Director of Certification

The CPC approved a job description for a program director. It was agreed that the program director will be hired by USENIX but does not need to be seated in the Berkeley office. It will be up to the program director to decide where support staff work. Eventually a centralized office will need to be established. In other work in the areas of leadership, governance, and management the subcommittee presented a revised organizational chart which we amended and approved. Below is a summary of the job description. Inquiries about this position should be directed to Gale Berkowitz at <*gale@usenix.org*>.

> "Supervises and directs the activities of all project staff and consultants, and coordinates the activities of volunteer leadership. Implements directives of the SAGE Certification Board. Is accountable to the SAGE Certification Board and the SAGE Executive Committee. Recommends and participates in formulating organization policies to achieve the goals established for the organization. Enforces administration of policies. Recommends and participates in planning immediate and long-term goals. Makes decisions appropriate to the implementation and execution of organization projects. Plans, manages, supervises, and directs all project functions. Develops policies and procedures for the project staff and determines goals and objectives for the project staff. Hires and evaluates project staff. Manages all contracts with consultants and testing organizations. Develops and administers annual project budget. Monitors fiscal expenditures of the organization and its committees. Serves as the Liaison between the project, its committees, SAGE, and other organizations with related interests."

Business Plan

We briefly discussed the executive summary of the business plan drafted by Mark Stingley and accepted it. This four-page document will be used in the fundraising and marketing efforts and is now available on the SAGE Certification Web site at

<*http://www.usenix.org/sage/cert/business_plan.html*>.

The Funding and Patronage Subcommittee will continue to contact the potential patrons and the committee was urged to provide him warm contacts.

In discussions with the Architecture Subcommittee we decided we are going to rely on the SAGE Job Descriptions for the description of the certification levels. There will be exam prerequisites, but they will not be verified for the first level of the exam.

With the Accreditation and Education Subcommittee we determined that the purpose of accreditation is quality control. Accredited institutions must have instructors with degrees in relevant areas, have appropriate experience teaching in the relevant areas, and the curriculum/teaching materials must be submitted to the SAGE Certification Program for review and approval. It was decided not to work with the universities on the development of training in the short term, but rather target the smaller training/learning centers.

The Marketing Committee's current focus is reviewing and recommending appropriate marketing firms.

The Administration and Procedures Subcommittee has been addressing applications, appeals, reinstatements, legal compliance, examination delivery,

certification acknowledgment, and operations maintenance. The program director will implement the policies and procedures. A first draft of the policies and procedures for the administration of the program is well underway.

Ethics and Discipline Subcommittee is working on a draft policy document (a draft presented in December suffered from scope creep). The plan is to rely on the SAGE Code of Ethics The committee is continuing to develop the consequences of fraud and cheating.

For more information about the SAGE Certification Project, please visit <*http://www.usenix.org/sage/cert/*>.

SAGE Certification Policy Committee hard at work. More photos can be found at <http://www.deas.harvard.edu/~lois/SageCertMeeting2.01/>.

USENIX news

Good Works

by Daniel Geer

President, USENIX Board of Directors

<*geer@usenix.org*>

Ignoring such cultural icons as The Grinch, who can argue with Good Works? If you have no idea that USENIX has a Good Works side, perhaps it would be best to first read <*http://www.usenix.org/about/goodworks.html*> and then we'll chat.

Assuming you are aware of what is on the above page, let me tell you some more. As I long since learned to recite, most important things are not seductive while most seductive things are not important. Candidate Good Works that are important are, in fact, harder to find than candidate Good Works that are seductive. This is where the Grinch comes in, and for the better.

We get proposals that are, charitably, charity cases. We get proposals that simply represent a few peoples' life work and dedication. We get proposals for startlingly fundamental research. We get proposals for operating expenses of very nice people shopping their desires door to door looking for "yes." We get proposals that are to play the winner and so deliver the next generation of contributors to our field. We get proposals that shore up orphan facilities that we all nevertheless depend on. How to choose? How to measure Good Works, and how to learn from that measurement?

First, as former Treasurer, I can tell you that USENIX is in a very sound financial position and that this was no accident. It is the result of a decade of husbandry, of living within a little less than our means, of getting a balance sheet together that permits a broader latitude for USENIX to experiment, to take risks. In all the business world, it is the balance sheet that buffers risk and bounds the risk-to-reward ratio, at least until it can't do it anymore. USENIX is in a favorable position for risk taking because we have been prudent about taking our own risk and balancing it with our purpose to help individual risk takers show what they can do, show what they have done.

Second, the monies that USENIX has are almost entirely derived from its attendees and exhibitors, that is to say from its members. We have had some gifts, but our favorable position should be thought of as the harvested proof that we are doing something right. So long as we can keep doing the right thing and the economy doesn't tank, this equilibrium can be maintained, modulo our essential need to track the march of technical progress and its steady wind of obsolescence.

That said, in what sorts of Good Works should we be engaged? Board members are called upon to vote money to Good Works every time we meet. Remember, this is/was your money we are talking about and Board members of any organization like this one are formally obligated to be prudent, forward thinking, and dedicated in their investment of the organization's funds and to the furtherance of their organization's goals. What should we do? By what outcome measure should we be judged, should our Good Works be judged?

The naive decision is simply "This applicant seems deserving; we have the cash; let's feel good by giving him some." We are not naive, but that is one end of the spectrum. The other end is more like "Is there any appreciable bang for the buck in this proposal that's out to get our members' monies?" In between is the dif-

ficult "What is the most strategic thing we can do for our members that except for this proposal will never get done?" and the even more difficult "Even if this is not our responsibility, is it nevertheless essential to our ability to keep getting our work done?"

Honorable people can and will differ on this. Speaking as President, my position, subject to new and better evidence from any quarter, is this: The mission of USENIX in <*http://www.usenix.org/about*> is well stated where it enumerates

- problem-solving with a practical bias,
- fostering innovation and research that works,
- communicating rapidly the results of both research and innovation,
- providing a neutral forum for the exercise of critical thought and the airing of technical issues.

It is that mission that rules. As I am absolutely convinced that USENIX is the very best organization that the computer systems community has, I take it as a responsibility to use whatever surplus we might enjoy beyond prudent reserves to advance that explicit mission on the grounds that we are the best there is to do so. I take it as a responsibility to spend the monies contributed by our members on our members, which isn't about buying steak dinners or subsidizing other less able organizations however seductively appealing they may be; it is simply our responsibility to invest in your capabilities, your thought leadership, your continuing capacity to evolve. It is not our responsibility to make charitable decisions for you – you can do that well enough on your own time and to your own taste.

You're right; my position has an edge to it. I have the single most skeptical multi-year voting record on proposals for Good Works, which coupled with a personal record of initiating more risky new venues than anyone else is at least consistent in its dedication to spending every dime of capacity we can generate on doing what we do well and on whom we derived it from, i.e., you and in proportion to your capacity to give something back of consequence. I like positive feedback loops. We are damned lucky enough to have this one.

Board Meeting Summary

by Gale Berkowitz
Deputy Executive Director
<gale@usenix.org>
and Ellie Young
Executive Director
<ellie@usenix.org>

The following is a summary of some of the actions taken by the USENIX Board of Directors between August 2000 and January 2001.

Conference Registration Fees for 2001

Conference registration and tutorial fees will be increased by $10 per day and the charge for registering after the cut-off date was raised to $100. Conference registration fees for the Annual Linux Showcase will increase by $100. Student fees will not change. Membership dues will remain the same.

Standards

The proposal from Stoughton for Standards work for 2001 was approved in the amount of $50,250. This year the principal area of focus in Standards work will continue to be with the "Austin Group," revising the POSIX and Single UNIX specifications. The resulting standard will replace the current ISO 9945-1 and 9945-2, IEEE 1003.1 and 1003.2, and The Open Group's SUS (XSH, XCU, and XBD). The Open Group continues to be an active focus for the new Open Systems standards. USENIX continues to play a critical role in the development of these standards.

Good Works

The Board voted to allocate $25,000 for the Computing Research Association's Committee on the Status of Women in Computing Research for the Distributed Mentor Project (<*http://www.cra.org/Activities/craw/dmp/index.html*>), in which outstanding female undergraduates work with female faculty mentors for a summer of research at the mentor's institution.

USENIX will again sponsor the USA Computing Olympiad (the annual computing competition for high school students) in the amount of $51,200.

A proposal by Lesley University, in collaboration with Polytechnic University, for $50,000 in funding for support of its Computer Clubhouse Network was approved. The aim of this project is to provide computer mentoring and training for underserved children at an after-school learning center.

USENIX will fund the Berkeley Foundation for Opportunities in Info Technology (BFOIT) (<*http://www.bfoit.org/*>) in the amount of $15,000 to increase representation among students of color in the computer-related studies at U.C. Berkeley.

USENIX voted to sponsor the HAL 2001 conference in the Netherlands from August 10–12 2001, in the amount of $10,000. This three-day, open air, networking event will focus on computer security, privacy, citizen rights, biotechnology, and other controversial issues affecting society as a whole.

SAGE Certification

The USENIX Board voted to offer at least $75,000 in matching funds for SAGE Certification. This is a challenge grant to other potential patrons of the Certification project. For more information about SAGE Certification Patronage, see <*http://www.usenix.org/sage/cert/patrons.html*>.

Conferences

BSD Con. USENIX will be taking over sponsorship of this conference in 2002.

AFS. A distributed file system workshop will be held in conjunction with the Annual Technical conference.

NordU. USENIX will make available $15,000 to cover speakers' travel expenses for 2002.

GUADEC Conference (GNOME Users & Developers European Conference). USENIX gave a grant of $10,000 to support travel costs for speakers and some attendees.

OpenBSD's "!Crypto 2001" Summit. USENIX will give a grant of up to $5,000 for travel and expenses for some of the developers to attend.

SAGE

Over the past several months discussions have been taking place among the USENIX Board of Directors, the SAGE Executive Committee, and the membership concerning the potential restructuring of the relationship between USENIX and SAGE. An overview of the direction of SAGE, remarks from the President of the Board of USENIX, and the Discussion Points for the restructuring can be found on the Web at *<http://www.usenix.org/sage/restructuring/index.html>*.

A business plan will be submitted by the SAGE Executive Committee to USENIX in early February 2001.

Next Meeting

The next meeting of the Board of Directors will be held April 3, 2001, in Berkeley, CA.

Twenty Years Ago in USENIX and in UNIX

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

I thought I'd take a break and look at one single event each in USENIX and UNIX history and their prime movers.

USENIX

In 1981, Lou Katz left Columbia University and trekked westward to Berkeley. And with him moved the Association.

By then, the center of gravity had shifted: much of the work on UNIX was being done west of the Delaware River; and it was clear that most of the conference attendees were from Texas, Colorado, California, etc. Following Lou's move, USENIX set up a real office and (at long last) took on some employees. The result was amazing.

UNIX

Also in 1981, UniSoft, founded by Jeff Schreibman, brought out a port called UniPlus+, which was compatible with System III and (in 1993) was still compatible with System V.

Jeff was one of the Berkeley students in fall 1975 who helped Ken Thompson bring up Sixth Edition on the newly arrived 11/70 (the other was Bob Kridle who, in 1983, was one of the founders of mt Xinu).

The next summer, Jeff supervised Chuck Haley and Bill Joy as they installed the fixes from the "50 bugs" tape. Interestingly, it was Schreibman who ported Joy's changes to the size of the data blocks on the VAX-11/780 to the PDP-11/70. But by then he had founded UniSoft.

Jeff, wherever you are, your deeds are remembered.

(Incidentally, /usr/group was incorporated in 1981; but that story's in my *Quarter Century of UNIX*.)

USACO News

by Rob Kolstad
Editor
<kolstad@usenix.org>

The USA Computing Olympiad has held two contests so far this year, the fall and winter USACO Opens. Each contest had two divisions, one for those starting out and one that challenges world-class competitors.

The fall contest attracted 280 entrants from 30 countries. Vladimir Novakovski, an underclassman from Thomas Jefferson High School of Science and Technology in Virginia, achieved the only perfect score. Contestants from Vietnam earned four of the top six spots.

I analyzed the programming languages used in order to see the influence of the USA's AP computer science curriculum (the Green Division sports the more challenging problems):

GREEN DIVISION				ORANGE DIVISION			
Program	Subs	C	Pascal	Program	Subs	C	Pascal
amicbl	155	88	67	sort	66	44	22
enemy	83	41	42	crypt	54	36	18
infrnd	109	60	49	vhist	82	60	22
outfrnd	112	69	43	parktri	107	80	27

As you can see, just over half of the submissions in the Green Division are in C (excepting the enemy problem) and 2/3 to 3/4 of the entries in the Orange Division are in C.

The winter contest had the most entries ever in a non-end-of-year contest – 310 entrants from 28 countries:

United States: 126	Korea: 9	Slovakia: 3	Denmark: 1
Georgia: 45	Indonesia: 8	Slovenia: 3	Estonia: 1
Belarus: 34	Latvia: 7	Argentina: 2	Germany: 1
Vietnam: 14	Canada: 6		
Netherlands: 2	Lithuania: 1		
China: 13	Poland: 6		
Croatia: 2	Singapore: 1		
Bulgaria: 12	Yugoslavia: 5		
Greece: 2	South Africa: 1		
Colombia: 10	Kyrgyzstan: 4		
Romania: 2	Turkey: 1		

Four contestants achieved perfect scores: Jan Oravec from Slovakia, Reid Barton from the USA, Nguyen Viet Tien from Vietnam, and Nguyen Kinh Luan from Vietnam (currently residing in Singapore).

The five problems in this contest were very difficult, with a mean score of 316 points out of 1,000 possible and a variance of 304 points. The scoring was challenging as well; only nine test cases (out of approximately 50) separated those who scored 750 points from those who scored 1,000. You can see detailed results and analysis at *<http://ace.delos.com/WINTER01res.htm>*.

Like any world-class event, the world's best competitors can often perform feats that appear to be superhuman. Here's another amazing effort (in the orange division) from Matthew Watson of the USA. It solves the problem: "Find the last non-zero digit in *N* factorial." This pro-

gram runs in milliseconds even for huge values of *N*:

```
int   n[] = {1, 1, 2, 6, 4, 2, 2, 4, 2, 8};
int r (int x) {
      int   w[] = {6, 4};
      if (x >= 10)
         return ((r (x / 5) * w[(x / 10)
            % 2] * n[x % 10]) % 10);
      else
         return n[x];
}
void main () {
      ifstream in ("fact.in");
      ofstream out ("fact.out");
      int     num;
      in >> num;
      out << r (num) << endl;
      in.close ();
      out.close ();
      exit (0);
}
```

Do you know a pre-college student that might like to compete in computer programming at the national or world level (this year's big trip is to Finland)? Please direct them to <*http://www.usaco.org*> so they can get signed up for our spring and US Open contests later this year.

Here's a challenging problem because its many solutions naturally break the programs into speed categories. It was problem 5 in the winter contest. Five-second time limit on a Celeron 400; good solutions run in under 750 milliseconds.

Cows in Bed [Burch, 2001]

Farmer John has N (1 <= N <= 5,000) cows who sleep in stalls in a barn with K stalls numbered 0..K-1. The i-th cow has a unique brand that is a number Si (1 <= Si <= 1,000,000). Each cow knows where to sleep because she sleeps in stall number Si mod K. Of course, cows will never want to share a stall for sleeping.

Given a set of cows and their brands, determine the minimum K such that no two cows sleep in the same stall.

INPUT FORMAT:

```
* Line 1: One integer: N
* Lines 2..N+1: One integer that is
        a cow's brand
```

SAMPLE INPUT (file bed1.in):

```
 5
 4
 6
 9
10
13
```

OUTPUT FORMAT:

A single line with the minimum value of K on it. All legal input datasets can be solved within the allotted time.

SAMPLE OUTPUT (file bed1.out):

```
8
```

CONNECT WITH USENIX & SAGE

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 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:

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

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

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