inside:

CONFERENCE REPORTS
USENIX Symposium on Internet Technologies and Systems

SAGE NEWS & FEATURES
Managing Your Career Using the 80/20 Rule
On Reliability – Backups, Restores, and Recovery
Toolman Features Steve Kinzler

STANDARDS REPORTS
POSIX.1h
The Single UNIX Specification, version 2

BOOK REVIEWS
The Bookworm
Python, PPP, Intranets, and Time Management

USENIX NEWS
A Second Start for the NLnet Foundation
News from the PGP Key Signing Service
Twenty Years Ago in ;login:
K-12 Outreach and the MVHS

ANNOUNCEMENT
1998 USENIX Annual Technical Conference

features:

One ISP's Response to the Problem of Spam
by Nick Christenson and Dan Farmer

Spam and the Law
by Scott Hazen Mueller

Interview with Dr. Clair W. Goldsmith
by Rob Kolstad

Source Code UNIX for PCs
by Robert Gray

Musings
by Rik Farrow

Using Java
by Mark Sienkiewicz

The Webmaster
by Dave Taylor

Using C++ as a Better C
by Glen McCluskey
# Upcoming Events

## 4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program chair</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 27-30/98</td>
<td>Santa Fe, NM</td>
<td>Joe Sventek</td>
</tr>
</tbody>
</table>

## System Administration, Networking, & Security (SANS) Conference

Sponsored by the SANS Institute, co-sponsored by SAGE

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 9-15/98</td>
<td>Monterey, CA</td>
</tr>
</tbody>
</table>

## USENIX Annual Technical Conference

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program co-chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 15-19/98</td>
<td>New Orleans, LA</td>
<td>Fred Douglass, Program Chair Glen Cole &amp; Berry Kercheval, IT Coordinators Jon &quot;maddog&quot; Hall, Freenix Track Coordinator</td>
</tr>
</tbody>
</table>

## 2nd USENIX Windows NT Symposium

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program co-chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>August 3-5/98</td>
<td>Seattle, WA</td>
<td>Thorsten von Eicken &amp; Susan Owicki</td>
</tr>
</tbody>
</table>

## Large Installation System Administration of Windows NT Conference

Co-sponsored by USENIX and SAGE

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program co-chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>August 5-7/98</td>
<td>Seattle, WA</td>
<td>Remy Evard &amp; Ian Reddy</td>
</tr>
</tbody>
</table>

## 3rd USENIX Workshop on Electronic Commerce

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program co-chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>August 31-Sept. 3/98</td>
<td>Boston, MA</td>
<td>Bennet Yee, Program Chair Dan Geer, Public Key Infrastructure Session Coordinator</td>
</tr>
</tbody>
</table>

## 6th Annual Tcl/Tk Conference

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program co-chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 14-18/98</td>
<td>San Diego, CA</td>
<td>Don Libes &amp; Michael J. McLennan</td>
</tr>
</tbody>
</table>

## 1st International System Administration and Networking (SANE) Conference

Organized by NLUUG, cosponsored by USENIX and Stichting NLNet

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program co-chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 18-20/98</td>
<td>Maastricht, The Netherlands</td>
<td>Edwin Kremers &amp; Jan Christiaan van Winkel</td>
</tr>
</tbody>
</table>

## 12th Systems Administration Conference (LISA '98)

Co-sponsored by USENIX and SAGE

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program co-chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 6-11/98</td>
<td>Boston, MA</td>
<td>Kev Gittler &amp; Rob Kolstad, Program Co-Chairs Phil Scarr &amp; Pat Wilson, IT Coordinators</td>
</tr>
</tbody>
</table>

## 3rd Symposium on Operating Systems Design and Implementation

Co-sponsored by ACM SIGOPS and IEEE TCOS

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program co-chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 22-25/99</td>
<td>New Orleans, LA</td>
<td>Marge Seltzer &amp; Paul Leach</td>
</tr>
</tbody>
</table>

## Deadlines

<table>
<thead>
<tr>
<th>Event</th>
<th>Notification to Authors</th>
<th>Final Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper Submissions</td>
<td>April 9/98</td>
<td>July 28/98</td>
</tr>
<tr>
<td>Extended Abstracts</td>
<td>April 17/98</td>
<td>September 4/98</td>
</tr>
<tr>
<td>2nd USENIX Windows NT Symposium</td>
<td>June 18/98</td>
<td></td>
</tr>
<tr>
<td>Large Installation System Administration of Windows NT Conference</td>
<td>June 18/98</td>
<td></td>
</tr>
<tr>
<td>3rd USENIX Workshop on Electronic Commerce</td>
<td>April 17/98</td>
<td>July 21/98</td>
</tr>
<tr>
<td>4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)</td>
<td>April 27/98</td>
<td></td>
</tr>
<tr>
<td>System Administration, Networking, &amp; Security (SANS) Conference</td>
<td>May 11/98</td>
<td></td>
</tr>
<tr>
<td>USENIX Annual Technical Conference</td>
<td>May 1/98</td>
<td></td>
</tr>
<tr>
<td>2nd USENIX Windows NT Symposium</td>
<td>June 18/98</td>
<td></td>
</tr>
<tr>
<td>Large Installation System Administration of Windows NT Conference</td>
<td>June 23/98</td>
<td></td>
</tr>
<tr>
<td>6th Annual Tcl/Tk Conference</td>
<td>July 28/98</td>
<td></td>
</tr>
<tr>
<td>12th Systems Administration Conference (LISA '98)</td>
<td>December 6-11/98</td>
<td></td>
</tr>
</tbody>
</table>
2 IN THIS ISSUE . . .

LETTERS TO THE EDITOR
3 In Response
   from Lee Damon
3 Erratum

CONFERENCE REPORTS
4 USENIX Symposium on Internet Technologies and Systems

SAGE NEWS AND FEATURES
19 Managing Your Career Using the 80/20 Rule
   by Tina Darmohray
19 I Sense a Change
   by Hal Miller
21 Member Survey Results
   by Hal Miller
22 San Antonio Revisited:
   Five Years of SAGE
   by Pat Wilson
23 On Reliability – Backups, Restores, and Recovery
   by John Sellens
28 Toolman Features Steve Kinzler
   by Daniel E. Singer

FEATURES
32 From the Trenches: One ISP's Response to the Problem of Spam
   by Nick Christenson and Dan Farmer
47 Spam and the Law
   by Scott Hazen Mueller
50 Interview with Dr. Clair W. Goldsmith
   by Rob Kolstad
53 Source Code UNIX for PCs
   by Robert Gray
59 Musings
   by Rik Farrow
62 Using Java
   by Mark Siennkiewicz
66 The Webmaster
   by Dave Taylor
68 Using C++ as a Better C
   by Glen McCluskey

STANDARDS REPORTS
73 An Update on Standards Relevant to USENIX Members
   by Nicholas M. Stoughton

BOOK REVIEWS
78 The Bookworm
   by Peter H. Salus
80 Programming Python
   Reviewed by Terry Rooker
81 PPP Design and Debugging
   Reviewed by Chris Kotaridis
81 First Things First
   Reviewed by Kartik Subbarao
82 Intranet Security: Stories from the Trenches
   Reviewed by William S. Annis

USENIX NEWS
84 A Second Start for the NLnet Foundation
   by Teus Hagen, Frances Brazier, Wytze van der Raay, and Jos Alsters
85 News from the USENIX PGP Key Signing Service
   by Greg Rose
87 Twenty Years Ago in ;login:
   by Peter H. Salus
88 K-12 Outreach: MVHS and the Student Network
   by Matt Shibla

ANNOUNCEMENT
90 1998 USENIX Annual Technical Conference
96 ;mtd
   by Rob Kolstad

April 1998 ;login:
in this issue . . .

I'm not sure you have to go to the ends of the earth to get off the information superhighway, but USENIX Executive Director Ellie Young is way offline as I write this, on vacation hiking in Patagonia. Meanwhile, as the rest of us try to do our jobs, email messages with subject lines like “What's New, what's Hot for YOUR Business!!” land with annoying frequency in our mailboxes (two copies of that one were waiting for me this morning), and unwanted and sometimes unmentionable items pollute USENET News. It's enough to make anyone decide to hitch a ride to Patagonia.

Happily the war on spam is escalating, and two meaty articles in this issue report on it from the trenches and the front lines. Nick Christenson and Dan Farmer describe the arsenal of anti-spam techniques, technical and otherwise, that a major ISP has developed, and Scott Hazen Mueller educates us on spam and the law.

Internet topics are featured extensively in this issue. You'll find reports on the USENIX Symposium on Internet Technologies and Systems; the Toolman column features a nifty command-line tool that interacts with Web browsers; the Webmaster column tells how to add counters to existing CGI programs; and an interview with Dr. Clair Goldsmith of the University of Texas reveals the complexities of dealing with abuse of online services in an academic environment. On the email privacy front, Greg Rose delivers a detailed update on PGP. (And why not segue into a plug for ;login;’s online version? If you haven’t already discovered it, each issue from October 1997 onward can be found at <http://www.usenix.org/publications/login>. Articles go online about a month after publication.)

This issue also inaugurates a new regular feature, Bob Gray’s “Source Code UNIX for PCs.” This time the author provides the background you need to get started running source code UNIX on a PC, and explains why it’s a good idea to do it. This promises to be an enjoyable and practical series.

Rounding out the issue are Rik Farrow’s musings on wearable computers and corporate acquisitions, an “On Reliability” article on backup and recovery, columns on Java portability and on mixing C and C++ code, a bundle of book reviews, and standards reports that include a continued description of the Single UNIX Specification, version 2.

We hope you enjoy this large economy-sized issue of ;login;. Keep an eye out for a special issue on security that will be coming your way later this spring.
Lee Damon Responds

Dear Editor,

In the February, 1998 edition of login, two writers took exception to my IMHO column, “WWW(ther(ing) Internet” (December, 1997).

Dear Messrs. Maples and Williams, the point of my article was not to attack the unwashed masses being unleashed on the Internet. My point was to challenge the 90s fad of opening an ISP at the drop of a venture-capitalist hat.

When everyone was rushing to open video stores in the 80s, there was no established community to be disrupted or destroyed. However, this is not the case with the ISPs that are popping up everywhere. The requirements to open an ISP (a bit of money, one person with half a clue on how to put the equipment together, a few phone lines) make it too easy for ISPs to open with insufficient training for staff, and no education for the users.

It is these video-store ISPs that I object to. Their entire goal in life is to make a quick buck, and be-damned to anyone they may hurt in the process. “If Spamford wants to send mail to everyone on the net, hey, it’s money in the pocket.”

The VS-ISPs unleash floods of people on the Net, but don’t bother to do anything to teach the users about the evils of spam, or how to be good Net citizens. These VS-ISPs do their users a disservice, as the users don’t get a chance to discover the entirety of the Internet. It is this lack of depth that will bore and drive people away, back to their TVs.

I can’t wait for the video-store ISPs to go away. That doesn’t mean I want their users to do so. As people gain experience and clues about what else is out there and about how to interact with others online, they will discover an entire new world of “Internet.” That will be good for all of us.

Lee Damon

Erratum

An incorrect URL for the Net BSD Project was printed on the inside back cover of the February, 1998 issue of login.

The correct URL is <www.netbsd.org>.
In April of 1982, DDN was mandated as the network of choice for data communications, receiving large amounts of federal funding. Although a network was being designed, the driving idea was clearly the protocols used by that network. If you could put these protocols in the hands of the users, demand would drive the military, the government, and commerce in the right direction. It would be a user-driven network. As a result, in January of 1983, the TCP/IP stack, the stack that DDN was using, was also mandated as the protocol stack of choice. The specifications called for high (99%) availability and low end-to-end delays.

At that point, ARPANET and MINET had been combined. In 1983, they were split into a government-driven component (operated as DDN) and the rest, which was called ARPNET.

In 1984, the DDN topology was pretty large, running throughout the US. Its biggest circuit had a bandwidth of 56 Kbps. In Europe, 9.6 and 2.4 Kbps were still the best bandwidth available, although connectivity was quite good.

As increased computer power was becoming more available and affordable; T1 (1.544 Mbps) lines started getting bought. Then Mosaic came.

At the same time (from the mid 80s to mid 90s), war was waged between TCP/IP and ISO's OSI. Many sides were trying to have TCP/IP established, and, in the end OSI died a slow death.

By the end of the 1980s, the 1984 deregulation was raging. It seems it will take a long time, though. Now, 14 years later, it's still just catching on.

The ingredients for the success of the Internet are:

1. It's user driven. The commercial products have to be released and working fast. Having it driven by the government would allow for longer delays.
2. It doesn’t bill according to time and distance of operations. Users pay for the size of their connections and have access to everybody else in the world. The economics of this billing model worked, and commercial enterprises learned how to be profitable from it. We are now trying desperately to hang onto that.

3. It was built on open systems and interconnected existing networks. First it mainly consisted of TCP/IP on top of X25. Now it’s TCP/IP on top of Ethernet or ATM.

4. It more or less stuck to a single standard (TCP/IP) everywhere, and a lot of work from many people went into it. It’s the one set of protocols that everybody is using, and we are not sure where we would be without TCP/IP as the one and only protocol stack.

The speaker then spoke of the present state of UUNET. Because UUNET’s was one of the core components of the public Internet, one can gain a good idea of what the Internet looks like right now.

UUNET is a large wholesaler. It provides a large backbone, to which other Internet service providers connect. Its core business is direct access to the Internet. Resellers offer dialup access and Web services.

UUNET’s DS-3 network was connecting Washington state, California, Texas, North Carolina, New York, and the Washington DC area in March of 1996. It seems its design had problems, though, because a year later demand increased by 1000% and the existing infrastructure couldn’t keep up. Most large wholesalers have actually started running out of bandwidth and can’t buy enough to cover the demand. At the same time, the prices have remained stationary. “All you can eat for $19.95.” The price is right, so the Internet is flourishing. At this time, 1.5 DS3 connections are installed per day.

UUNET was bought out twice, first by MFS and then by WorldCom. This was fortunate, because UUNET was now paired with the facilities providers. They could actually tell them what they needed worked out, twice a year, and have them do it. WorldCom in fact more than doubled the size of their network, from 11,000 miles to 25,000 miles. The two busiest areas were Silicon Valley and Washington, DC, as well as east-west connections.

Latency has always been the big issue. The objective was to get the latency down to 100ms. This is still the objective.

The persisting fact is that there is a tenfold increase of demand per year. UUNET’s backbone demand doubles every four months, and by December 1999, the network will be (at least) 1,000 times larger. Still, 20% of UUNET’s sites have latency higher than 100ms. At the same time, there is an OC12 network overlayed on top of the DS3s, with plans to move on to OC48s in the next few months. As the speaker put it, “If you’re not scared by all this, you don’t understand what’s going on.”

The new routers installed are given a year. After that, nobody knows what is going to happen. In other words, the state of the current network is one year ahead, and if no significant improvements are made in a year, sales will have to stop. Five-year planning is a joke.

In the more global view, Europe is three to four years behind (mostly E1/T1 links), but has already been deregulated. Market penetration is not that high yet (only 5% of European homes have a PC), but Europe is growing fast. The Pacific Rim is three to four years behind Europe.

On the metropolitan dial-in front, the Microsoft Network is a venture funded by Microsoft, but owned by UUNET. It involved about 300,000 modems at the time of the symposium (within the DOW network), with large increases planned. Its design goal is to obliterates blockage.

The future is unclear. The speaker argued that we’re not in the middle or the end of this race; we’re at the beginning. Commercial companies are still learning how to deploy this technology and be able to pay for it.

Heiden predicted that FAX is going to be the next major Internet application (late 1990s). It’s now 50% of all transAtlantic communications. Next year it is going to be on the Internet, incurring 5,000,000 sessions a day on dial-in networks. He also suggested that IP voice will be the next big Internet application after that, while system design (of Intranets/Extranets) is going to become more active. Boiled down, the argument rests between packet-switched and circuit-switched technology.

The speaker identified the next wave of challenges as:

- regulatory (applications): IP voice, international regulation
- infrastructure distribution: Intranets and Extranets
- technology: really fast switches and routers

Concluding, Heiden predicted that the next five years will be very exciting and neurotic. Anyone whose business lies within this technology will do well.
**TECHNICAL SESSIONS**

**Session: Caching I**  
Summaries by Mark Mellis

**Study of Piggyback Cache Validation for Proxy Caches in the World Wide Web**  
Balichander Krishnamurthy, AT&T Labs – Research and Craig E. Wills, Worcester Polytechnic Institute

Several factors need be considered in the construction of Web caches – size of the cache, i.e. the amount of main memory and disk space allocated to it; replacement policy, i.e. choosing which valid data items to keep in the cache, and coherency policy, i.e. ensuring that the data in the cache is consistent with the data on the server. Current Web cache implementations typically use time-to-live (TTL) techniques to maintain cache coherency. Objects that don’t have explicit expiration times are flushed from the cache after a time that may be fixed or determined heuristically. The subject of the authors’ research, Piggyback Cache Validation (PCV), maintains cache coherency by piggybacking: whenever a cache communicates with a server, it adds validation requests for potentially stale objects “on the back of” the message. In this way, the number of cache-server messages is kept low while the coherency of the cache is improved.

As is the case in much of the work presented at USITS 97, this research was conducted by simulating the performance of a Web proxy cache using large traces of actual proxy traffic. The authors’ PCV work showed a 16-17% reduction in number of cache-server messages, 6-8% reduction in average cost, and 57-65% reduction in cache staleness ratio, all in comparison to the best TTL-based policy. Krishnamurthy indicated that there is more to come along this line of research – he considered this paper to be the “least publishable unit.”

**Exploring the Bounds of Web Latency Reduction from Caching and Prefetching**  
Thomas M. Kroeger and Darrell D.E. Long, University of California, Santa Cruz, and Jeffrey C. Mogul, Digital Equipment Corporation

When examining approaches to optimizing systems, one of the first questions a designer asks is, “Where are the biggest wins, and how big are they?” By developing a sense for the bounds of performance in the ideal case, the designer decides upon implementations that best meet his or her goals. Kroeger presented research examining performance bounds for latency reduction using two techniques: caching and prefetching. Since the point of the research was to find bounds, the authors looked at optimal caches – those with infinite size and with complete knowledge of future events. Using these optimal caches (limited in various ways), they used trace-driven simulations to determine that in the caching-only model, latency could be reduced at best by 26%, while in the prefetching-only model, latency could be reduced by at best 57%. A model that used both caching and prefetching could reduce latency by at best 60%.

The principal factor limiting the potential performance improvement of caching was found to be the rapid turnover of information on the Web. The distance into the future that a prefetching cache can predict is a principle factor in its ability to limit latency, with four minutes of prescience being enough to produce substantial performance improvement.

**The Measured Access Characteristics of World Wide Web Client Proxy Caches**  
Brad Duska, David Marwood, and Michael J. Feeley, University of British Columbia

David Marwood presented research results that helped provide a baseline for intuition: what do real client access patterns look like and what implications do those have on existing cache performance?

Using both trace-driven simulation and static analysis, the authors examined some 47 million requests made by nearly 24,000 clients in seven discrete data sets, preserving the privacy of the clients by passing the traces through a one-way function that preserved the uniqueness of each client while obscuring its identity.

Marwood showed that with second-level cache sizes ranging from 2 to 10 gigabytes, hit rates between 24% and 45% can be expected. Between 20% and 70% of accesses are “false misses” caused by weaknesses in the squid and CERN cache coherence algorithms. Sharing is bimodal between widely shared and narrowly shared objects, and widely shared objects also tend to be shared by clients from different traces (everyone reads Dilbert . . .).

More information on the work described in this presentation can be found at <http://www.cs.ubc.ca/spider/marwood/Projects/SPA>

**Session: Servers**  
Summaries by Mark Mellis

**A Highly Scalable Electronic Mail Service Using Open Systems**  
Nick Christianson, Tim Bosserman, and David Beckmeyer, Earthlink

The rise of national Internet service providers has created needs for traditional computing services delivered on scales far larger than those previously contemplated. Christianson presented the email architecture used by Earthlink Network, Inc., which currently accommodates
more than 400,000 users with over 560,000 mailboxes. The system throughput approximates 13 million messages per week, with 40 POP sessions initiated per second and 600 active POP daemon processes running at any one time. The email system has a 99.9% uptime record. The architecture described is expected to scale to greater than one million users.

Prime criteria in the design were message integrity, general robustness (uptime), scalability, performance, cost effectiveness, and legacy considerations. The system is designed around four main functional areas.

Front end servers receive inbound SMTP messages from the Internet. These are stock Unix machines running unmodified sendmail. Local delivery is via a custom mail.local program that interacts with the authentication database and delivers messages to mailboxes stored on Network Appliance file servers.

POP servers are another set of machines that handle interaction with subscribers and delivery of subscriber messages to the Internet. The POP servers also interact with the authentication server and mount user mailboxes via NFS.

Mailbox storage is handled by Network Appliance fileservers. Mailboxes are split over several servers and their location is computed by a balanced hash over 319 directories and stored in the authentication database.

Authentication service evolved from standard UNIX password files, through newdb databases, and is now managed by a commercial database product and accessed via SQL. With 400,000 users, it is no longer possible to assign each user a unique UID in the traditional UNIX sense. The same authentication database serves email, access servers, Usenet news, and other services.

One of the most significant challenges has been in file locking, necessary in mail delivery. Few commercial systems have lock tables large enough or lock table lookups fast enough for Earthlink’s needs, so they are currently using file system semaphores for locking. This remains an area of ongoing development.

**Improving Web Server Performance by Caching Dynamic Data**

Arun Iyengar and Jim Challenger, IBM T.J. Watson Research Center

Much work has been reported in the area of client caching static Web pages. This presentation by Arun Iyengar in contrast reports on server-side caching techniques that are well-suited for dynamic content. Dynamic content delivery is often up to two orders of magnitude slower than static content delivery. By caching dynamic pages at the server, substantial performance increases can be gained. The author described the DynamicWeb cache, used by IBM in support of the 1996 Atlantic Olympic Games Web site, where it achieved a hit rate of around 80%. The DynamicWeb cache is part of the forthcoming net.data offering from IBM.

Because of the transient nature of dynamic Web content, explicit communication between the application and a process known as the cache manager is required to ensure that only long-lived content is cached. IBM has developed an explicit application program interface (API) for application-cache manager communication. Although the cache manager can run on the same host as the application, it most often is deployed on a separate host. Furthermore, multiple cache managers can interact with a single application, and a single cache manager can manage several caches. This allows the system designer great flexibility in his task.

DynamicWeb can satisfy several hundred requests for dynamic content per second given typical workloads, and in tests displayed near-optimal performance in many cases and 58% of optimal performance in the worst case.

**Measuring the Capacity of a Web Server**

Gaurav Banga and Peter Druschel, Rice University

Banga’s presentation described limitations in current Web server benchmarking methodologies and presented a method for generating synthetic Web server workload that more closely resembles real life and can economically produce traffic volumes large enough to overload even high capacity servers. He noted that in benchmarks such as WebStone and SPECWeb96, the clients stress the server under test by operating in lock step, with only a single outstanding request per client. Reaching rates of 1100 requests/sec in this scenario requires the use of 74,000 client processes.

The authors developed S-Clients (short for Scalable Clients) to enable them to generate enough workload to produce request rates similar to those “observed in nature,” on an affordable number of client computers. Rather than following the structure of a traditional http client, an S-Client is designed to shorten TCP’s connection timeout, and to maintain a constant number of unconnected sockets that are trying to establish new connections. These design goals allow S-Clients to saturate the server with a small number of clients and to ensure that the generated connection attempt rate is independent of the rate at which the server can accept new connections.
Banga displayed graphs illustrating the measured degradation in performance of a Web server when pushed into overload by S-Clients, for both steady-state workloads and bursty workloads. While previous test methodologies showed flat server performance once maximum capacity was reached, the S-Client methodology showed marked performance degradation as presented workload increased. During the question and answer session following the formal presentation, a conference attendee from CNN commented that he had observed similar performance degradation in real life.

More information on this work, including source code, can be found at <http://www.cs.rice.edu/CS/Systems/Web-measurement/>.

Session: Potpourri
Summaries by Petros Maniatis
BIT: A Tool for Instrumenting Java Bytecodes
Han Bok Lee and Benjamin G. Zorn, University of Colorado, Boulder

The objective of this work is to characterize the behavior of Java programs, using instrumentation of Java bytecodes. Furthermore, an approach was taken so that the instrumentation tools produced would be easy to modify so that they meet users' changing needs.

Metainstrumentation is the process of creating instrumentation tools. BIT is the first metainstrumentation tool for Java. The idea behind it is to insert calls to methods to keep track of the number of instructions that get executed while the program is running. Its interface was modelled after ATOM.

BIT consists of a set of Java classes that can be used to build customized tools. It takes a class file, modifies it, and outputs another class file. The BIT system contains instrumentation and analysis code.

The instrumentation code in the libraries deals with modifying a class file and producing the output file. This code has been used to accomplish the motivating goals, namely, insertion of calls before each basic block, extraction of basic metrics per basic block, and maintenance of accumulated counters. Obviously, the instrumentation code shouldn't alter the behavior of the program (for instance, it shouldn't change the number of instructions that get executed per basic block).

The analysis code in the libraries specifies what to do when the methods inserted by the instrumentation code actually get invoked. In this instance of the problem, this code just keeps track of the number of instructions in the basic blocks.

BIT works by hierarchically breaking a class file into the following entities: the program, which is divided into methods, each of which is divided into basic blocks, each of which is divided into instructions. The current system allows navigation among these elements, information gathering, and insertion of calls into the analysis methods in the library.

BIT targets any language that targets the Java Virtual Machine (i.e., any language for which there exists a compiler that outputs Java bytecodes). It can be used to create simple customized tools such as:

- count the number of times a method is invoked
- count the number of bytes of code during the execution of a program
- measure the branch probability on a per-branch basis
- measure the instruction usage

Furthermore, BIT can be used within other, more advanced tools:
- profiling (flat or hierarchical)
- calling context trees
- program optimizations
- reorganization
- JIT optimization via annotation and relation to hardware coverage analysis
- branch prediction

The performance of BIT has been evaluated on two sample tools produced with BIT:

- a branch counting tool
- a dynamic instruction counting tool

Five Java applications have been used as input to these tools (one of which was BIT itself). The execution times of the instrumented applications were increased by 1.1 to 2.5 times (compared to their execution times before instrumentation). The execution time increase is mainly due to the fact that calls to the analysis methods have to be executed, in addition to the normal code that constitutes the program. The code sizes of the instrumented applications were increased by 1.1 to 1.4 times (compared to their code sizes before instrumentation), for obvious reasons.

HPP: HTML Macroreprocessing to Support Dynamic Document Caching
Fred Douglish, AT&T Labs – Research, Antonio Haro, College of Computing, Georgia Institute of Technology, and Michael Rabinovich, AT&T Labs – Research

Dynamic Web pages, especially those produced by search engines, usually have a low percentage of dynamic resources. In other words, the dynamic data in them are usually far smaller in size than the static data, such as formatting, banners, ads, etc. In some cases, such pages display extensive repetitiveness in their structure. Many attempts have been made to take
advantage of these characteristics in order to reduce the amount of bits transmitted per such page.

Data compression and delta encoding have been proposed and implemented in the past. Both approaches result in an increase in the amount of work that the Web server has to do. A different approach was introduced with the OBJECT tag in recent HTML specifications, whereby dynamic objects are embedded in a page. The browser then fetches these dynamic objects separately, which causes an excessive number of requests to be received by the server. Especially in the case of search engines, devising templates containing OBJECT tags to accommodate an average random query can be difficult and inefficient.

The approach taken by the authors of this paper extends HTML with macro-instructions. They provide named placeholders within a cacheable template, which the client fetches first. Then the dynamic data are retrieved separately in a single request. Finally, the dynamic data retrieved are placed within the template and presented to the user.

Similar work has been done with SHTML (server side includes) and ASP (active server pages). The argument the authors provide in favor of their system is that, unlike SHTML and ASP, it executes the expansion of the templates at the client, offloading the server significantly and therefore improving the performance of the server, which is the hot spot for heavily used services.

In summary, the benefits of this work include:

- Bandwidth savings (by not sending multiple copies of similar pages)
- Server load reduction
- Template pre-compression (since templates are static and can be cached/compressed)

- Some help with TCP slow start (by overlapping the calculation of the dynamic data at the server with the transmission of the template)

One of the more powerful aspects of this approach is the use of loops to accommodate multiple similar fragments of a resource, as needed in the case of search engines. Nested loops are also supported. Furthermore, conditional macroexpansion and assignment to macrovariables have been included.

In terms of performance, the most striking result is that the latency observed using this system is only slightly worse than delta encoding. However, the comparison was done between a prototype implementation of this system and the fastest known delta-encoding implementation. Moreover, the extra effort is spent at the client, thereby allowing the server to perform other work at the same time.

**Session: Security**

Summaries by Petros Maniatis

**Lightweight Security Primitives for E-Commerce**

Yossi Matias, Alain Mayer, and Avi Silberschatz, Bell Laboratories, Lucent Technologies

The objective of this work is to provide a secure mechanism to support electronic commerce that works well with micro-transactions and in a mobile environment. This has been accomplished using a client-side proxy service.

The basic motivation was drawn from the emergence and popularity of personal-ized electronic commerce applications on the Internet that supply a very large number of very low cost transactions (like a stock quote ticker or an online stockbroker). Such applications normally maintain a long-term relationship between the server and each client (in the previous example, the stock market service provider and the ticker software at the client's computer), normally in the form of a subscription. Every time new quotes arrive or the user asks “How much did I make this morning?” the client has to pay for the service. Such services have, individually, very low monetary value. The effort devoted to authenticating such a low-value transaction shouldn’t be disproportionate.

Currently, SSL (the Secure Sockets Layer) is the predominant method of authentication on the Web. It can run on top of any TCP connection and has very wide applicability. Other such mechanisms with similar characteristics are S-HTTP (Secure HyperText Transport Protocol) or SET (Secure Electronic Transactions). However, all these methods are much more elaborate (and expensive) than the average microtransaction, like the one mentioned previously; they do not provide any flexibility with respect to cost and complexity. For instance, SSL needs to do a public key encryption/decryption per transaction, which is quite expensive. Furthermore, users have to maintain states about their subscriptions on a specific client host (in the form of cookies, for instance).

The proposed solution is modular. It supports secure subscription and initialization of information delivery, ensuring data privacy, authenticity, and integrity to a reasonable degree, as well as third-party provability (nonrepudiation).

Upon initialization of the subscription of the client to the services offered by the server, a single shared key is agreed upon. This is computed at the client's side, given the user's identity, a unique identifier for the service and a local secret. This
shared key is then encrypted using the server's public key and transmitted to it. The server maintains a record of the client's identity, along with the associated shared key. Notice that the public key operation just mentioned happens only once, when the subscription is initialized. Subsequent exchanges are encrypted using this single shared key.

Secret key computation on the client's side is a very important component of the system. The function used must be efficient, computable, and consistent across different clients (i.e., it shouldn't require memorization of any secret across sessions). It should also provide modular security – guessing a key (computed through this function) shouldn't provide any help to derive other keys. The function used in this work is the Janus function, used in Lucent's Personalized Web Assistant. This function is based on pseudorandom functions and collision-resistant hashing.

In terms of performance, an elaborate HTML page (about 10kb) takes about 0.06 seconds to be encrypted, whereas using RSA public key encryption of a single key takes about 0.12 seconds and a single RSA signature takes about 1 second. In other words, this system encrypts a whole page twice as fast as RSA encrypts a single symmetric key.

This is not supposed to be a standalone solution, especially because SSL already seems to be a standard. However, the results of this work are intended as a supplementary security mechanism, to be used along with SSL. In a typical scenario, the handshake between the client and the server would occur using SSL; subsequent communication could be performed without SSL.

For more information, the interested reader can look at the Lucent Personalized Web Assistant.

Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2

Li Gong, Marianne Mueller, Hemma Prafullchandra, Rolland Schemers, JavaSoft, Sun Microsystems, Inc.

The objective behind this work was the expansion of the security model present in the Java system, as found in the released Java Developers' Kits. The authors have produced a new, more flexible security model, allowing fine-grained access control of system resources.

In its past versions, the Java security model relied on the concept of the Sandbox. Local applications/applets were always trusted and were allowed to do whatever they pleased with system resources. Remote applets were always suspected for foul play and were allowed to run only within a very restricted environment (called Sandbox), without local filesystem or network access, to name a few of the restrictions. These were relaxed with the next version of the JDK, where key-signed applets (using certificates from a third party) could be trusted, even if they were downloaded from the network.

Both approaches are still too cumbersome for the requirements of contemporary Web-based services. For instance, a stock portfolio maintenance applet would need access to local financial records. However, there would be no need to allow such an applet to access other, unrelated files or to open arbitrary network connections. With JDK 1.0.x, such an applet would be unusable unless explicitly installed by the owner of the system. With JDK 1.1, this applet could be installed, if properly certified, but would have far greater privileges than those required to complete its purpose. Furthermore, local code cannot be unconditionally trusted by default, because a naive user could have inadvertently downloaded it from the net and run it.

The new Security Architecture (which is going to appear in JDK 1.2) implements a policy-neutral framework allowing multiple domains of protection to be imposed on any applet. The security policy configures a series of Sandboxes, depending on the code currently executing in the Java Virtual Machine. This new architecture does not have a built-in notion of trusted code, regardless of the code's origin (local or remote).

The basic components in this framework are:

- security policy
- typed permissions
- protection domains
- multidomain computation (trust among mutually suspicious parties)

Policies can be chosen per site, per user, or per application. Conceptually, they resemble a table indexed by code origin and credentials. Every entry in the table assigns a set of permissions to each origin and signature, although these can be overridden. Permission inference is also implemented, so rules can be used to cover cases not explicitly included in a policy. For instance, if the policy rule allows connecting to "any host in the domain A-Domain.com," the inference module will have to decide whether this applies to "A-Host.A-Domain.com." The reader is referred to the paper for details about protection domains and example scenarios where pieces of code from multiple domains are combined.

In terms of performance, each security call is estimated to take about 200-300 microseconds in midrange systems in the current implementation. Security
computations are evaluated in a “lazy” manner, i.e., nothing is done until a computation is required. The “eager” computation would mean that every time a domain boundary was crossed, the set of security permissions would have to be recomputed. Because domain crossings are more frequent than security computation requests, lazy evaluation is better for the average case.

This new Java security framework will be released soon, along with the rest of the new JDK version 1.2.

Secure Public Internet Access Handler (SPINACH)

Elliot Poger, Sun Microsystems Inc., and Mary G. Baker, Stanford University

The objective of this work is to provide an intermediate-grade, secure access mechanism for public network ports. The concept of a prisonwall is introduced, whereby unauthorized – or as yet unauthenticated – public port users are confined within a small, protected subnetwork. Only when proper access privileges are certified can they use local network resources and/or the Internet.

In most modern, technologically friendly buildings in the computing industry and elsewhere, network ports often appear in public lounges, hallways, or conference rooms. This is bound to become quite common soon in public libraries and educational institutions. These ports should be usable by all those having permission to use them through a permanent affiliation (as would be the case for students without a permanent office in their departmental building), or a temporary affiliation (as would be the case of visitors in a research facility).

It is imperative that access to public network ports is controlled. Their malicious use could embarrass the host organization (if, say, an unauthorized user initiated an attack against another organization from the host organization’s facilities), endanger local operations (if an unauthorized user mounted an attack against the host organization itself, taking advantage of local trust), or even cause the host organization to break contractual obligations (if an unauthorized user, taking advantage of network location, accessed data licensed only to persons directly affiliated with the host organization, like online encyclopedias and such).

The basic idea behind SPINACH and the prisonwall scheme relies on separating public ports from trusted ports (trusted ports would be those within offices and behind locked doors). These public ports are connected to the rest of the network through a “prisonwall” device, which provides a mechanism for users to become authorized. When users first connect to a public port, they are “imprisoned” within the prisonwall. They cannot send or receive packets whose other endpoint lies outside the prisonwall. After users are authenticated with the system, packet delivery works as usual with the outside world. Note that a prisonwall works like a firewall, but turned inside out.

The idea was put to use in the authors’ building (Computer Science Department, Stanford University). The design goals were:

- ease of deployment
- a system that works with current network infrastructure and existing client hardware and software
- no effect on trusted ports
- ease of use

- availability to both temporary users (visitors) and permanent users (local staff temporarily using a public port). (Authorized users should be allowed to use the network without restraint. The authorization process should be short and easy.)
- ease of administration
- maintenance of an audit trail for better damage control (Maintenance of the system should impose minimal administrative overhead.)

In this specific implementation, the prisonwall concept (called SPINACH) uses locally installed Virtual LAN (VLAN) switches to separate public ports from trusted ports into a “public subnet.” The SPINACH router selectively forwards traffic to and from the public subnet. It filters ingress traffic based on the hardware address and IP address of packets. It also performs the authorization/authentication functions for access control, using Kerberos.

An important decision that the designers of the system had to make was tradeoff security for flexibility and ease of deployment. SPINACH had to work with existing networking hardware on site, as well as widely available software on most users’ laptop computers (such as DHCP and Telnet clients). That limited the sophistication of security mechanisms they could use for the packet switching function at the SPINACH router. This is an issue because IP addresses and hardware Ethernet addresses can be spoofed without significant trouble. However, there are other options, which designers in other installations can elect to take.
Session: Monitoring
Summaries by Petros Maniatis

Web Facts and Fantasy
Stephen Manley, Network Appliance, and Margo Seltzer, Harvard University

The objective of this work was to provide a contemporary, broader characterization of Web servers on the Internet, proposing a taxonomy for them, and analyzing their access patterns, especially those related to growth.

So far, studies have focused on relatively shorter log traces of relatively narrower sets of servers. The authors managed to acquire 38 server logs spanning between one and two years, from multiple sites on the Internet, including Internet service providers (ISPs), universities, commercial sites, adult entertainment sites, free service sites (free software distribution), and informational/governmental sites.

The growth characteristics of all sites were astounding. Although not all growth was positive, growth patterns categorized sites according to the most highly correlated other observed factor. In all cases, however, growth was exponential (either increasing or decreasing).

The free software site was growing along with the number of Web users visiting it. Other sites in this category would be those requiring continual access (such as news feeds, as in ESPN or CNN).

The traditional business site was growing every time it underwent major renovation or redesign. The surges of accesses tend to decrease in time when the renovation becomes regular and expected.

Internet service providers and other sites hosting large numbers of user home pages were growing along with the number of user Web pages they contained (because each user tends to receive a close to constant amount of traffic). Such sites were the academic sites, the free Web-hosting sites like Geocities, etc.

Informational/governmental sites were growing along with the number of similar informational Web pages they could supply per user. Such sites tend to be indifferent to external traffic (and they happened to be the least popular).

Sites relying heavily on favorable treatment by search engines (such as the adult entertainment site) tend to grow along with their position in the result list of popular topics in popular search engines. The adult entertainment site observed had to shut down when its listing in most search engines stopped appearing close to the top.

Finally, sites accessed on a for-fee basis tend to grow inversely proportionally to the fees incurred per access. The organization site observed lost a lot of its traffic when the access price structure became less appealing to the public.

Another major concern on the Web involves latency. Users tend to abort a Web request after a while (estimated around ten seconds). Businesses geared toward Web advertisements are especially interested in decreasing the average latency for a page below the tolerance threshold of Internet users. Traditionally, the blame has been ascribed to the overloaded servers, to excessive Computer-Generated Interaction (CGI scripts), and to the concurrent maintenance of too many TCP connections.

Measurements were taken on location at the traced sites. Logs recorded processing latencies in most cases. Latencies were measured per byte. Most requests (around 90%) attained performance exceeding 1 millisecond per byte. There were, however, byte latencies reaching 100 milliseconds per byte.

As far as CGI-related latency was concerned, servers responded in less than one second to CGI-enabled requests. In most sites, even at different scales (from the most busy to the least busy servers), CGI was indistinguishable from non-CGI traffic and never surpassed 2% of the transferred bytes.

More work on the issue is under way, because it was concluded that the minute-long latencies observed could not be attributed to either excessive server load or high numbers of concurrent TCP connections between a server and its clients. This will mainly focus on a more detailed observation of heavily accessed Web servers so that the blame for portions of the measured latency can be assigned more accurately.

SPAND: Shared Passive Network Performance Discovery
Srinivasan Seshan, IBM T.J. Watson Research Center, Mark Stemm, and Randy H. Katz, Computer Science Division, University of California, Berkeley

The objective of this work was to provide a shared measurement scheme, offering information on expected performance characteristics of a network connection.

On the Internet today, we can find many clouds of local connectivity (which are, in fact, a Local Area Network in some incarnation or another) that can communicate with each other through a number of network hops. The basic problem is figuring out in advance what the net performance is going to be like when communicating with a host in a different domain (i.e., a different cloud).

The ability to change the content fidelity (quantity of information, smaller vs. larger images, lossy vs. lossless compression, etc.) according to the available network resources could be very useful. Such
would be the case if our Web browser were able to automatically turn off image loading for slow connections. Another case would be choosing among multiple sites mirroring the same information (similarly to Harvest). Finally, we can picture a Web page where every link is annotated with the expected bandwidth of the network connection it requires.

SPAND utilizes shared network performance measurements, which get propagated to nearby clients. Clients send summaries of their connectivity to the performance servers. The measurements themselves are passive; they are obtained by observing existing traffic on links—the only traffic produced by SPAND contains the performance measurement reports shared among neighboring clients.

SPAND has several components. SPAND-aware clients run modified applications, which can extract performance measurements from their existing, ongoing network connections. They subsequently make up performance reports, which they send to the performance servers. Servers receive summaries of connectivity by specific applications on local clients inside performance reports. They also receive performance report requests, to which they respond with aggregate reports (i.e., reports made up from all performance reports sent by local servers).

Packet capture hosts take on the task of creating performance reports when there are no SPAND-aware clients available. They snoop local traffic and generate performance reports on behalf of unmodified clients. Packet capture hosts are very important, because they make the rapid deployment of the system much easier. This is because they can, for the short term, substitute the existence of modified applications on local clients.

An important issue is that performance measurements are categorized per application. Each application has its own traffic characteristics (in terms of transfer type, bulky or interactive, flow control, congestion control, reliability, etc.), so performance measurements are kept separate per application type.

In practice, the authors have implemented a prototypical Web proxy that uses the SPAND toolkit and appends appropriate icons to links on a Web page to indicate expected performance behavior. The server's turnaround time is very promising, according to the results provided. The warm-up of the performance database is fairly fast. At first, the server can respond to a performance request 70% of the time. Eventually, this scales up to 95%. Also the measurements of the accuracy of the supplied estimates are very promising.

Future work includes the incorporation of better, more elaborate methods to derive aggregate performance reports, utilization of locality information to infer performance expectations from other nearby destinations, and protection from erratic measurements.

**Rate of Change and other Metrics: a Live Study of the World Wide Web**

Fred Douglass, Anja Feldmann, Balachander Krishnamurthy, AT&T Labs - Research, and Jeffrey Mogul, Digital Equipment Corporation - Western Research Laboratory

The objective of this work is to characterize change in the World Wide Web and to use this characterization to evaluate the benefits of rigorous Web caching.

Web caches are rapidly gaining popularity and attracting bigger research efforts (Squid/ Harvest, NetaCache, Cisco Cache Engine, Inktomi Traffic Server, and so on). It is still uncertain, however, how well they work. The point behind their use is to serve pages to users without having to contact the content provider, thereby eliminating unnecessary network traffic. In the same vein, delta encoding, a scheme according to which only changes to a Web page are transmitted and applied to a previous, base version, is gaining popularity as well. One of the questions this work is attempting to answer is "How often will you see changes that you can apply to the base version?"

To answer many similar questions, the authors studied large logs of Web traffic, to which they applied many metrics, in order to characterize different aspects of change on the World Wide Web. The metrics they actually used are:

- frequency of reaccess (it affects cacheability directly.)
- rate of change (the fraction of accesses that involved changed resources, which affects both cacheability and the applicability of delta encoding)
- age of resources and modification intervals (This helps to detect modification patterns, which affect how expiration times are selected on caches.)
- duplication of content (mirrors, aliases)
- changes in semantically meaningful ways (phones, embedded URLs)

A few of the basic facts retrieved from this study were that:

- The more frequently accessed resources change more often (stock tickers are a good example of that).
- Images are more static.
- Resource size is relatively unimportant.
- Commercial sites have more dynamic content.
The data used for this study comprised primarily a 17-day log of port 80 (the HTTP port) between AT&T Labs – Research and the world (which contained 465 clients and 21,000 servers and reconstructed the full content of all requests and responses). A second trace, taken at Digital’s WRL site, covering two days of Web traffic, supported the results extracted from the first trace.

The change ratio metric indicates the fraction of the total accesses that are, in fact, accesses to changed data. Evaluation of this metric over the two traces showed that images very rarely change. The most rapidly changing resources are octet streams (netcast streams like tickers). HTML is bimodal. About 70% of the HTML resources never change, whereas most of the rest of their remainder change on each access.

Age analysis of the data showed that the most frequently accessed resources are, in fact, the youngest ones. In general, HTML resources are younger than GIFs, and smaller images seem to run older than larger ones. Finally, educational sites tend to be the “oldest.”

Another issue the authors addressed was that of duplication. There are several causes for duplicated content:

- different URLs applied to the same resource (because of session IDs, appearing as parameters to a CGI script)
- mirroring of entire sets of resources
- duplication of icons or other images (like the Netscape Navigator logo)
- multiple URLs belonging to the same Web advertisement
- the rate of duplication being surprisingly high (around 18%)

Finally, the authors studied changes in semantically identifiable portions of retrieved resources, like images, email, or telephone numbers. They parsed the textual resources in the logs and found that HREFs appeared in 74%, IMGs in 72%, and email and telephone numbers in 20%. Email addresses and telephone numbers don’t seem to change much, as expected. HREFs and IMGs were unchanged in about half of the examined resources, but changed completely in 3-5%.

In summary, this study showed, through the analysis of data coming from two major organizations, that frequent changes happen in many resources. This might make delta encoding more useful than simple caching for those cases. However, duplication was very common as well. Therefore, the Distribution and Replication Protocol proposed by Marimba and Netscape might also prove useful.

**Session: Applications**

**Summaries by Mark Mellis**

**RainMan: A Workflow System for the Internet**

Santanu Paul, Edwin Park, and Jarir Chaar, IBM T.J. Watson Research Center

In these days of strategic partnerships and geographically dispersed organizations, systems that enable and manage distributed workload and preserve process are increasingly important. Paul presented RainMan, a distributed workflow system built on the RainMaker workflow framework.

RainMan is a loosely coupled system of network connected performers. These performers handle tasks in response to service requests generated by workflow sources. Performers can be individual humans, computer applications, or other organizations. RainMan provides mechanisms for managing performer work lists, a directory service, and a variety of user interfaces that enhance its use in cross-organizational environments, including the ability to manage shared work lists.

Paul indicated that a publicly accessible Web page describing RainMan was under construction. Contact him at <santanu@watson.ibm.com> for more information.

**Salamander: A Push-Based Distribution Substrate for Internet Applications**

G. Robert Malan, Farnam Jahanian, and Sushila Subramanian, University of Michigan

Salamander is a publish/subscribe data dissemination substrate for Internet applications. In use for several years supporting distributed research efforts, it offers a variety of services to its client applications.

Describing one of the principle Salamander applications, Malan told of the Upper Atmospheric Research Collaboratory (UARC) project, where groups of space scientists all over North America and Europe collaborate as frequently as daily on campaigns in real-time over the Internet. UARC’s goal for Salamander was to replicate the feel for the participants of working together in “a hut in Greenland,” providing access to instrumentation, informal “chat” communications, a shared annotation database, shared text editing, and most importantly, real-time access to experimental data.

The Salamander architecture includes a channel subscription service, application-level quality of service policy support, and a lightweight data persistence facility employing caching and archival mechanisms.

More information on Salamander, including source code, is available at <http://www.eecs.umich.edu/~malan/salamander>.
Creating a Personal Web Notebook
Judi Manber, University of Arizona

Manber, a self-described “academic that hacks,” is the author of the popular programs glimpse and agrep. Here he presents Nabbit, a tool for the construction of a personal Web notebook.

Two existing mechanisms for preserving “precious nuggets of information” from the Web have well-known shortcomings. Adding URLs to a hot list soon results in an unmanageable hot list, while saving entire Web pages locally yields overflowing disks filled with stale data in files with forgettable names. Hence, Nabbit. Nabbit is a WYCIWYG tool—What You Click is What You Get. It allows one to select sections of Web pages and paste them into a notebook, complete with time stamp and clickable reference to the original source. It even works with forms—a search engine form collected via Nabbit is still functional in the notebook.

Nabbit works by using approximate string matching to locate the selected information within the document source. Approximate string matching is what Manber does—“when you have a hammer . . .”) Once the selection is located, only the tags relevant to the selection are extracted, forming a stand-alone snippet of HTML. For this reason, Nabbit only works on HTML pages, not on Javascript. UNIX and Windows versions of Nabbit are available, though the UNIX version is more functional. More information about Nabbit is available at <http://glimpse.cs.arizona.edu/udm.html>.

Session: Works in Progress
Summaries by Mark Mellis

ScalaServer
Vivek Pai <vivek@cs.rice.edu>

Pai described research into system-level support for scalable network servers using commodity hardware and software. This support includes lightweight IO mechanisms, network stack enhancements, cluster support, and content-based request distribution. More information is available at <http://www.cs.rice.edu/cs/systems/ScalaServer>.

Why We Wait
Jeff Mogul <mogul@pa.dec.com>

Mogul discussed his investigation of Web latency based upon network packet traces rather than proxy logs. Examining one real-time hour of data—approximately 2 million connections—he found that the predominant cause of latency was lost SYN packets. He attributed the worst latency problems to network stacks that failed to include modifications to mediate SYN flooding attacks.

Data Collection with Mobile Agents
Jeremy Hylton <jeremy@cnri.reston.va.us>

The goal of Hylton’s research is efficient application-specific data collection. In his approach he pushes the indexing code onto the server. More information is available at <http://www.cnri.reston.va.us/home/koel>.

Evaluating High Performance APIs in AIX
Erich Nahun

Nahun observed that Microsoft achieves significant performance improvements by exploiting particular APIs in its Internet Information Server (IIS) on Windows NT. His research is focused upon exploring the performance benefits of similar APIs in non-Microsoft environments. His approach is to implement the APIs as kernel extensions and then modify servers to use them. Preliminary work shows improvements in process-based servers and the potential for even greater improvements in threads-based servers.

Workstation Authorization
Peter Honeyman <honey@citl.umn.edu>

Honeyman reported on work with goals similar to those of the SPINACH project reported upon in the Security session—allowing controlled, authenticated access to campus networks via Ethernet for mobile user communities. Honeyman’s approach differs from that of SPINACH by requiring special software support on the client computers, and by permitting access by actively controlling the ports on the ethernet hub.

Wisconsin Proxy Benchmark
Pei Cao <cao@cs.wisc.edu>

Cao described her work on a benchmark that uses LAN-connected clients running a trace-derived workload to measure latency and throughput in Web proxies. More information is available at <http://www.cs.wisc.edu/wisweb>.

Lucent Personalized Web Assistant
Alain Mayer <alain@bell-labs.com>

This work allows one to use personalized Web sites securely and privately while protecting against SPAM-oriented address harvesting and without having to remember a unique password for each site. These goals are achieved by accessing the personalized site via a proxy that mediates requests and manages authentication while protecting the user’s identity. The proxy does not maintain per-user state—it computes the necessary information using cryptographic techniques. The service can be tested at <http://lpwa.com:8000>.
Session: Caching II
Summaries by Mark Mellis

**Cost-Aware WWW Proxy Caching Algorithms**
Pei Cao, University of Wisconsin, Madison, and Sandy Irani, University of California, Irvine

Cao discussed issues influencing selection of Web cache object replacement algorithms, reviewed a number of extant strategies, and introduced a new algorithm, GreedyDual-Size (GD-Size).

GD-Size is a variation on one of a range of algorithms originally proposed by Neal Young. It is implemented by assigning a value \( H \), which is a function of the cost to obtain the object and the size of the object, to each object in the cache, and maintaining the \( H \) values in a priority queue. At each object replacement, the object with the lowest \( H \) value is removed and the remaining \( H \) values are reduced.

The authors demonstrated that GD-Size outperforms other widely-used replacement algorithms in hit ratios, latency reduction, and network cost reduction. More information on GD-Size is available at [http://www.cs.wisc.edu/wisweb/](http://www.cs.wisc.edu/wisweb/).

**System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace**
Steven D. Gribble and Eric A. Brewer, Univeristy of California, Berkeley

Gribble and Brewer performed a 45-day long packet trace of the network connecting a University of California, Berkeley modem pool to the Internet. The trace included some 24 million requests from six thousand clients. Gribble presented the results of the analysis of that trace. Unlike other work that has been reported upon in the past, this research was performed from a client perspective rather than a server perspective, and analyzed network traffic rather than using server or proxy logs.

The client community consisted mostly of Berkeley students and displayed strong geographic locality, which contributed to the strong diurnal cycle observed in the traffic patterns. Although peak-to-average traffic ratios of 5 to 1 were common on time scales of tens of seconds, the authors did not observe the self-similarity of network traffic described in other studies. This observation was the subject of lively discussion during the question and answer session.

A particular area of interest to the authors was diversity in Web browser client software. They observed 166 discrete clients, varying in type, version, OS, and hardware platform, although 55% of the browsers were Netscape Navigator on Windows95, with Netscape Navigator on MacOS the second most prevalent at about 20%. Considering volume of traffic, the authors observed that 31% of the bytes transferred were JPEGs, 27% were GIFs, and 18% were HTML, with 24% other types.

The traces generated during this study are available from the Internet Traffic Archives at [http://www.acm.org/sigcomm/ITA/](http://www.acm.org/sigcomm/ITA/), and more information on this study is available at [http://www.cs.berkeley.edu/~gribble/papers/papers.html](http://www.cs.berkeley.edu/~gribble/papers/papers.html).

Session: Information Retrieval and Searching
Summaries by Petros Maniatis

**The Search Broker**
Udi Manber and Peter A. Bigon, Department of Computer Science, The University of Arizona

The objective of this work was the creation of a librarian of online libraries in a form similar to a search engine. According to the author, search engines are very good already, although there is room for improvement. However, they
ack focus. Even when the user is looking for an item within a particular category, search engines return Web pages as results. For instance, if the user is looking for an x-ray of a shoulder, the average search engine will return Web pages containing the key words “x-ray” and/or “shoulder,” perhaps close to one another. Still, those returned pages will not very often contain an actual shoulder x-ray. In other words, key word searching is not good for all purposes.

The Search Broker attempts to capitalize on the widespread existence of specialized databases on the Internet. Several hundred such databases, covering quite specialized subjects, from x-rays to hotels to language dictionaries, have been found, connected, and searched through an integrated front end. At the time of the presentation, the Search Broker contained 412 subjects.

The model of the query fits into a two-level paradigm. First the appropriate database is located; this is done by matching the first word against a list of key words identifying topics. The database associated with the topic identified is then used to answer the specific question. For example, to look for a recipe with lentils, a user would have to submit the query “recipe lentils”; “recipe” selects a Web-based recipe database, and then “lentils” is the actual text of the query submitted to the database. If a topic cannot be identified, a regular search engine is used.

Other examples of queries would be “stocks IBM,” to find the current price of IBM stock, “fly tus jfk” to find a flight travelling today from Tucson, AZ to New York City, and “patent Manber” to find all the patents filed whose holder is someone called Manber.

The subject databases are added into the system by a human librarian. This was a decision intended to assure content quality. However, the addition of a new subject is fairly easy and quick (about one minute per new database). Once the URL of a new database is located, a utility reads its front end (assuming it is a form-based interface) and stores a representation for it, assigning default values to secondary fields. Then the librarian has to characterize the database (assign it a topic/category) and add links to related topics.

For more information and to use the actual system, interested readers can look at the Search Broker page.

**Using the Structure of HTML Documents to Improve Retrieval**

Michal Cutler, Yungming Shih, and Weiyi Meng, Department of Computer Science, State University of New York, Binghamton

The objective of this work was the improvement of the information retrieval techniques used for the World Wide Web, using HTML structural information.

Most automatically populated search engines (unlike those requiring manual classification of URLs) attempt to match a user query to their set of indexed documents. The documents matching the query are ranked according to certain suitability criteria. In most cases, the rank is a measure of the overlap between the concepts of the query and the document text. The main idea in this work is to use hypertext and HTML tag information to improve the retrieval results for such queries.

This is done by associating key words within a document with a classification vector, which contains the number of occurrences of the key word in any one of the predefined structural classes. Such classes can be hyperlink anchors, headers, titles, emphasized text, etc. Through this association, the appearance of a requested key word in an “important” structural class (like the title) of a document can reinforce our belief in the relevance of the document to the query containing the key word. Assigning appropriate weights to these structural classes was an important part of this work.

A technique used to provide better insight to the contents of a Web page is to see how other documents cite it. When we provide a link to a Web page, we try to supply a good description of it. The authors use such descriptions (also referred to as “anchor descriptions”) to improve the relevance of the contained key words. Furthermore, anchor descriptions can provide alternate wordings (in the form of synonyms or related terms) of the concepts contained in the cited Web page, thereby alleviating the “language variability problem.” Anchor descriptions constituted one of the structural classes mentioned previously. The complete list of structural classes used is:

- title
- top headers (Headers 1 and 2)
- bottom headers (Headers 3 through 6)
- strong (strong, emphasized, underscored, lists)
- plain (none of the above)
- anchor
To prove the concept, the authors built a search engine, called Webor (for Web-based search tool for Organization Retrieval) that indexed the entire Web space of SUNY at Binghamton and computed the term frequency vector per structural class for all key words in all pages. Given ten user queries and manually selecting the most relevant pages among all those indexed, they used hill climbing to derive the weights for the six structural classes (called "class importance values").

The metrics used to evaluate the results were recall and precision, established metrics in the field of information retrieval. They placed the greater significance on precision, because it measures the proportion of "good" documents out of all those retrieved. For the best values found at the time of the presentation (1 for plain text, 1 for bottom headers, 8 for strong and anchor, 6 for top headers, and 4 for title), they managed to improve the average precision by as much as 44% (for a 5-point average).

In the future, the authors plan to use more than ten queries to refine the class importance values derived, use larger page collections, and reevaluate the classes themselves (perhaps more/other classes would give better results).

SASE: Implementation of a Compressed Text Search Engine
Srividhi Varadarajan and Tzi-cker Chiuhe, Department of Computer Science, State University of New York, Stony Brook

The objective of this work is to combine searching and text compression in the same efficient framework.

The motivation behind SASE lies within current key word searching mechanisms used in most popular search engines.

Efficiency tends to become a problem, as the Internet grows. Being able to search through compressed text files without decompressing them first would provide increased flexibility and improved performance characteristics. As with most integration schemes, such an approach has to deal with tradeoffs, because it needs to achieve relative efficiency in both component subsystems (compression and searching).

In the text searching field, the most common approach uses inverted indices. Such an index records the location of every word in a database. It contains a dictionary of all the words, along with a linked list per word of all its locations in the database. When the user enters a query, the word is found in the dictionary, and its locations are retrieved through the linked list.

In the text compression field, the most popular approaches use substitution of repetitive patterns with shorter numerical identifiers. Variable bit length schemes have been used (Huffman codes), as well as dictionary-based schemes (Lempel-Ziv).

For the purposes of this work, an inverted index with a dictionary-based compression scheme has been used. Furthermore, the same dictionary used for the inverted index is used for the compression. The compression granularity is a single word, reducing compression efficiency, but allowing for searches that target word-based patterns.

Inverted indices tend to be very large, customarily about three times the size of the original database. One way to reduce the inverted index size is to introduce the concept of blocking into the system. The compressed word stream is partitioned into blocks. Pointer lists in the dictionary then point to these blocks, as opposed to individual words. During search, a retrieved block is searched linearly (in compressed form) for the exact location of the word in question. The block size determines the balance in the tradeoff between search time and storage space.

The system has been implemented as a stateless server with a Java front end. State is maintained by the clients (through a "context," which operates like an iterator: along with every response, the server returns the context, indicating which ones among a list of multiple responses it just returned. If the client needs more, it resubmits the query along with the context.).

SASE has been evaluated against GZIP (one of the best lossless compression utilities) and GLIMPSE (which also performs compressed text searching) on three databases:

- stories from Project Gutenberg (7MB)
- Internet RFC database (70MB)
- USENET news articles (300MB)

The compression efficiency of GZIP is 7-17% better than SASE, which in turn is 28-31% better than GLIMPSE. In terms of its search performance, searches take between 13 and 120ms, which compares favorably to a fully inverted index scheme. Search times increase linearly with the block size.

Future work plans include incorporating SASE in a news server (NNTP) to avoid transition overheads, using vantage point trees for approximate searching, and making SASE updatable (to avoid unnecessary reconstruction of the indices).
Managing Your Career Using the 80/20 Rule

by Tina Darmohray

Tina Darmohray, editor of SAGE News & Features, is a consultant in the area of Internet firewalls and network connections, and frequently gives tutorials on those subjects. She was a founding member of SAGE.

I get tired of repeating something I’ve already done. For me, the “fun part” of any day is the time that I spend on figuring things out. Basically, I like problem solving; give me something that doesn’t work (yet) and some time, and I’m happy trying to figure it out, configure it, and get it to work. It was like that when I was in school, too. I preferred problem set assignments to labs, reports, and lectures. I think it’s because I like to see quick results or at least feel I’m actively working toward them. In a work environment, this translates to avoiding meetings. Somehow I never feel like anything gets “done” in them, and I find that frustrating. I also love to teach. I like to watch the “lightbulbs go on” on the faces of the attendees; it’s cool.

Maybe I’ve just rattled off more specific preferences about how I spend my working day than you normally might expect to hear. (Maybe you’d add that I must innately enjoy analyzing things!) But I’ve developed this list over the years, out of necessity. I use it as a career management tool. I apply it to try to ensure that I get to do or learn what I’m interested in. It keeps me from taking jobs that aren’t a “good fit.”

I had worked as a UNIX system administrator the first four years out of college, when I took a position as an SE (what the company called their presales technical support engineers). According to the feedback I got, I was good at it. Over time, I realized that I really just didn’t like the job content. I liked the people, and the challenge was OK, but I just couldn’t find the little “problem sets” to solve on a daily basis. The bottom line was that 80% of what the job required wasn’t what I really liked to do. I was living for that 20%, which usually took a backseat to the other tasks I had on my plate. When an old colleague called me up and offered me a large network (complete with an Internet connection) to manage, I realized that I couldn’t pass it up.

I learned an important lesson through that brief career digression, which I’ve refined since then: it’s important to manage your career so that you get to do what you’re interested in! Sound simple? It can be. You just need to measure each opportunity you have against what you want to do. Including what you want to be doing in the future. That way, you guarantee that you’re spending most of your time on things that you like best or preparing for the job you want to become qualified for.

Start your own career management by creating a list of the things you like to work on. Include things that you don’t now but want to learn. Then apply the 80/20 rule to every new opportunity. Break the prospective new job down into its job content. Don’t glorify any aspect; if anything, be more critical than usual. This should give you a realistic list of what tasks make up the job. Now step back. What would you be spending 80% of your time on if you took that job? For example, if you really want to write code for a living, take a help desk job where “you’re also encouraged to create shell scripts to enhance your productivity” wouldn’t be the job for you. You need a job where 80% of it is writing code and 20% is interacting with users, not the other way around.

This measurement becomes even more important when you realize that, in every job, there will be about 20% of it that you “never get to.” As likely as not, in the previous example, the users are going to keep you too busy for you to ever write a script! So remember, for a job to be a good fit, the things you want to do should fall into the 80% area of what it really takes to do that job.

I’ve found that systematically measuring every new job using my 80/20 rule makes career decisions easier and more clear cut. And, probably most importantly, I make better decisions. I might turn down more opportunities by religiously using this process (because it doesn’t allow you to “read into” any position something that’s not really going to be there), but at least I don’t wind up headed in the wrong direction.

Try it for yourself and let me know what you think.

I Sense a Change

by Hal Miller

President, SAGE STG Executive Committee
<halm@usenix.org>

It seems to me that we have just entered a new phase in the development of our community. I haven’t yet been able to pin down what, but I think we need to start contemplating what changes are coming.

There was a time when most system administrators were trained in other professions and somehow fell into sysadmin work. Now more and more of us were hired into explicitly system administration positions, formally trained in some sort of computer science. Most of us began in relatively homogeneous environments (perhaps VMS and one or two flavors of UNIX). Most environments are now highly heterogeneous (multiple UNIX versions, Microsoft OSEs, Macs, on varying platforms). We used to be all customer support; now we have “backroom” folks as well as “front line.” System
administrators used to be all "jacks of all trades" but are now more and more becoming specialists.

Perhaps our field is expanding (additional operating systems and platforms). Maybe the demands placed upon us are increasing as users begin to redirect the tasks they do into our domain. The proliferation of PCs without tools to support them on the network has changed how we break down our workday.

I have also noticed the change at LISA. We used to see new tools and developments presented. Most of the "new" now tends to be enhancements of existing tools or procedures — very few actual new developments are appearing (maybe we are just too busy to develop?). Those who used to attend the sessions are spending more time sitting outside the sessions now. Although sessions are still well attended, most attendees are the "newer" members. Perhaps we are getting bored? Perhaps there is a dichotomy developing: haves and have-nots? Or have we just taken all the tutorials and sessions that are practical for experienced sysadmins?

I suspect these (how we are made up and what is happening at LISA) are related. Where are we going? It's too early to say. But it seems evident to me that "stay the course" will not be a viable option much longer. We need to read the signs and begin to interpret them, so we can start taking steps to implement new goals.

Here are some possible paths:

- Experience-level split. We could see services and conferences become geared to only the more junior or the more senior types. We could have multitrack conferences, with tracks aimed toward only junior or senior folks.
- Speciality breakouts. We have UNIX-only folks. We are now picking up NT-only people. We have front-line and backroom, systems and networks, commercial/government/education/research breakouts. All of these have, of course, lots of big similarities, or we would not be in one organization.

However, many of these have big differences. We now have some specialty workshops. Do we need to change our emphasis from one big organization into a conglomerate of small ones?

- New development funding. We have begun funding "good works" projects, where we find deserving sysadmin causes and help get them under way. We could begin a larger program of projects to spur technical development.
- Education, mentoring, and apprenticeship. We want to do this anyway, so should these be the only applications of SAGE resources? Should we become a service provider?
- Staying the course. If things are OK now, maybe they'll stay that way. If we have fully and properly identified what we do and where we're going, perhaps we need not change anything.

Here are some things to consider: with regard to the experience-level split, many thoughts occur to me, mostly negative. One of the biggest advantages I see to LISA is the mentoring that occurs by mixing junior and senior folks. A relatively "limited" percentage of people are willing to identify themselves as "junior," thus possibly diluting any benefits to be gained. I don't like the idea of enforcing such a dichotomy. For the same reasoning (1 think we need to keep close to each other rather than split up), I shy away from too many "breakouts." We need to maintain one organization, one set of goals, etc.

I don't like the idea of changing the business we are in (providing for the profession and its members). I'm all in favor of increasing our involvement in the advancement of the education process and think it is crucial to our survival, but I don't want to see it become a business. We need to help coordinate and assist, not provide training directly.

I do like the idea of using our money to advance the technical side of our business. SAGE has been working to advance our organization, education, and professionalism, and this provides an opportu-

---

SAGE, the System Administrators Guild, is a Special Technical Group within USENIX. It is organized to advance the status of computer system administration as a profession, establish standards of professional excellence and recognize those who attain them, develop guidelines for improving the technical and managerial capabilities of members of the profession, and promote activities that advance the state of the art or the community.

To achieve its mission SAGE may:

- Sponsor technical conferences and workshops;
- Publish a newsletter, and/or professional short topics series;
- Develop curriculum recommendations and support education endeavors;
- Develop a process for the certification of professional system administrators;
- Recognize system administrators who are outstanding or are otherwise deserving of recognition for service to the professional community;
- Speak for the concerns of members to the media and make public statements on issues related to system administration;
- Promote and support the creation and activities of regional or local professional system administrators.

SAGE STG EXECUTIVE COMMITTEE

President:
Hal Miller <halm@usenix.org>

Secretary:
Tim Gassaway <gassaway@usenix.org>

Treasurer:
Barb Dijker <barb@usenix.org>

Members:
Helen Harrison <helen@usenix.org>
Amy Kreiling <amy@usenix.org>
Kim Trudel <kim@mit.edu>
Pat Wilson <paw@usenix.org>
nity to fill another need. Alone, it doesn’t solve the problems I pose in this message, but it does help keep us going in the “right” direction.

These are merely some observations and thoughts. I see changes beginning to happen to us, and I want to get an understanding of what this means before it creates a problem. I think it’s clear that SAGE is not “done” with its work, but what that work will be in the future isn’t clear. I think we should not fragment ourselves into small topical groups, but we need to begin addressing a wider range of needs. It’s hard to have the “right” answers before you understand changes that are just beginning to become evident. I hope some of you will have some observations and will write some “letters to the editor.”

Member Survey Results

by Hal Miller
President, SAGE STG Executive Committee
<halm@usenix.org>

In early January I emailed a survey to sage-members@usenix.org, the mailing list that covers the largest number of SAGE members in one “place.” The purpose was to get some feedback on some of the more controversial issues SAGE faces so the executive committee would have a reasonable chance of acting in the manner most closely aligned with the majority of the membership. Here are some results and comments.

I asked whether you would like to see certification of sysadmins. Three to one said yes, but about one-third of the total were undecided. When asked what form of certification you would prefer, you responded with a strong plurality in favor of a combination of single topics and a more comprehensive plan, with a large number selecting only single topic. Combining the two, over half appear interested in the single topic form. Combining the “combination” with the “comprehensive” preferred categories, nearly half support some large-scale plan. To the question regarding the relative importance of certification, you seem to see it as something between “unimportant” and “essential,” roughly split in the middle of the spectrum.

Many people have requested that SAGE supply an email forwarding service. In answering the question on this point, two to one said yes you’d like it, but about half were undecided. The split on whether or not you would subscribe to such a service was similar.

I asked about SAGE’s involvement in the formal standards arena (e.g., POSIX, IPv6, etc.). Three-quarters of you rated this “very important,” with most of the remainder giving it the lesser rating that was still “important.” Very few rated it “unimportant.”

Three-quarters of you “always” read login. Just over half say that the articles are “sometimes” helpful or worth reading, with all the rest saying it is “always” good.

I described the Short Topics series of booklets as reference material on not necessarily technical issues and asked whether more technical booklets should be published in the series. Less than a third said we should do this, with most of the rest saying that other methods of publication were better for this. (I happen to agree).

I proposed a new series of “How-To Notes,” to be a one- to two-page (Web-accessed) checklist or summary of the basics of subjects (e.g., configuring ssh, dealing with purchasing, etc.) There was very strong support from you on this idea (three-quarters). I asked for suggested topics and received a long list with lots of repeats (to help us prioritize!)

The last question was on your view of the Code of Ethics. Eighty percent feel that it
should be maintained or elevated in status, with only one respondent saying it should be given decreased emphasis.

We as a board were pleasantly surprised by the high level of response and by the positive attitude shown in the accompanying comments (all of which I have handy to refer to).

So what does it mean? Our intended actions are as follows. We are drafting a proposal for a combined program of single-topic and baseline-competency certification. We intend to ensure that it remain strictly voluntary and that sysadmins not be required to pay large sums to private companies to "buy" a certificate. We will ensure that certification maintains a high standard and real value. I hope a working proposal/plan/test scenario will be made public during this calendar year.

On email forwarding, the large number of undecided, the large number who said they would not make use of it, the number of alternatives available, the cost (setup and recurring), and problems associated (spam) have convinced us to put our resources elsewhere this year. The issue is not fully "dead," but I don't anticipate action on it in the foreseeable future.

There had been discussion about getting SAGE out of the standards business. We just reversed that and will instead increase our role this year.

The response regarding ;login: is heartening. We will try to increase our use of it as a method of communication. A second note here: this is obviously a good place for you to make your views or technical advances known!

The Short Topics series will continue as is. We have four booklets in the pipeline at the moment and (thanks to the survey respondents) a long list of topics to search for authors to cover.

Expect to see "How-To Notes" details soon.

Other things are also happening in SAGE—we aren't limited to this list and will keep you posted.

Start planning now for LISA. Write a paper! Oh, and if you missed the survey, you might consider again the idea of subscribing to <sage-members>. It's a majordomo list at <usenix.org>.

---

San Antonio Revisited: Five Years of SAGE

by Pat Wilson
Pat Wilson is a member of the SAGE STG Executive Committee and is an invited Talks Coordinator for LISA XII.

<paw@northstar.dartmouth.edu>

As I write this, I'm just returning from the USENIX Security Conference in San Antonio. The last time USENIX was here (the general conference in June of 1992), postconference events included the launching of SAGE. It's certainly been an interesting five and a half years.

Who could have predicted, five years ago, that the demand for sysadmins (not to mention the recognition that system administration is something worth hiring people to do) would be so high? The current wealth of opportunity is almost obscene; everywhere you look, installations are growing, whether it's moving from a glass-house mainframe shop culture to distributed desktop computing or back the other way, rediscovering the economies of scale of centralized systems. It adds up to one thing — more technology to manage and (hopefully) more folks to manage it.

What we considered a "large" system five years ago is now at best middle sized. The Internet, that vast suck of bandwidth, is now a commodity item. There are ads for computer chips on TV. Everyone has email. There may be slightly fewer versions of UNIX out there (depending on how you count), but there's no one clear market leader. There's more to know, more to do, and (thanks to "Web time") less time to do it in.

And what of SAGE? Starting from a handful of people in a room in this hotel, we now have well over 3,500 members (and over half of all USENIX members also belong to SAGE). We co-sponsor three conferences (LISA, SANS, and LISA/NT), publish a series of Short Topics booklets (the fourth of which, on hiring sysadmins, should be in members' hands soon after you read this), have drafted a Code of Ethics, and have seen (and in many cases assisted in) the growth of several SAGE local groups (currently 17 scattered around the US).

Thanks in large part to the early codification of the SAGE jobs descriptions and the publication of the booklet, many organizations not only have a better understanding of the sysadmin role, but some have constructed career paths for system administrators. Degree programs with emphasis on system administration are beginning to appear at some colleges, and efforts to grow sysadmins in high school have begun. There's certainly lots yet to do regarding coordinating education and evaluation efforts, but we're on the right path.

Where will SAGE be in another five years? Although no one (that I know, at least) has an infallible crystal ball, it's hard to believe that there won't be systems to administer. The problems we solve every day, after all, remain essentially the same despite the underlying technology. I've yet to run across vendors whose ideas on how to use their products completely match the reality of my environment, or products that completely protect the end-users from themselves. Wherever sysadmins go, SAGE will be there.
On Reliability – Backups, Restores, and Recovery

Way back in the first article in this series (login: vol. 22, no. 3, June 1997), I mentioned some “general principles” for reliability. One of those general principles was “plan for failure and recovery,” and that’s going to be the primary focus of this article. More specifically, I’m going to be talking about backups, restores, and recovery, with the focus on how you deal with your data, rather than on which data you need to back up.

This paragraph is, of course, where I remind you of the basic tenets of reliability: service levels, risk evaluation, costs of failures, appropriateness for your environment, etc. One nice thing about this article’s focus is that almost everyone will agree on the necessity for proper backups, so your justification document/business case may be a much easier sell this time.

Let’s define what we (well, I, actually) mean by backups, restores, and recovery. In this article I’m going to be concentrating on data backups to some secondary storage medium. The most common medium for doing backups is magnetic tape (which comes in a wide variety of types), but some situations call for alternatives, such as regular old hard disk (or DASD for you big iron fans), floppy disks (and their removable media cousins, such as Iomega’s Zip and Jaz units), CD-ROMs, optical disks, paper tape, and punched cards (though the latter two have fallen somewhat out of favor in recent years). A “restore” is the process of retrieving a file (or a few files) from the backup media. And “recovery” is what happens when something goes very wrong (fire, flood, earthquake, theft, presidential scandal, etc.) and you need to put everything back in order.

I’ll also mention “archival storage.” An archive is very much like a backup, except that it’s intended to be kept for the long term, perhaps forever. Again, the medium used varies, depending on cost, security, and retrieval considerations. I’ve mentioned “archival storage” primarily so that you’ll know what I’m not talking about.

I should mention, just for the record, why it’s important to do backups. In a sense, it’s not the backups that are important; it’s the restores and recoveries. Most people have had the experience of accidentally deleting or corrupting an important file, and most system administrators have been faced with users who have also done just that. And it’s not just human error that can corrupt files – as hard as it may be to believe, some software actually does have bugs. Although disks are more reliable now than in the past, sooner or later a disk is going to fail, burn, or get stolen, and you’re going to need to undertake a disk recovery. Think of backups as insurance – the mere presence of reliable backups not only prevents your disks from failing; reliable backups also keep you from getting fired the next time there’s a flood in your machine room.

What’s important about backups? Backups should be current, consistent, and complete. Current and complete are easy to define – you need to do backups on a regular and reliable schedule (usually daily), and you need to back up all the files and directories that you intended to back up. Consistency is a little less obvious – your backup system should be able to deal with files that are changing during a backup. An easy example of a changing file is a large file used for database storage, which could easily change between the time you start reading the file to back it up and the time you’ve finished reading the file, thereby giving you a nice looking but useless backup. Backups also need to be available when you need them – more on that later. (For a good discussion of var-
Whether you use a home-brewed shell script, some freely available software, or an expensive commercial product, your goals are the same: get the backup done, and get it done right.

ious backup-related issues, see [1] in the LISA V proceedings, which also contains a number of other backup-related papers.)

One of our reliability tools is "automation," and backups are a perfect task to be automated – they're done regularly, they're boring (but necessary), and they're very important. Whether you use a home-brewed shell script, some freely available software, or an expensive commercial product, your goals are the same: get the backup done, and get it done right. Let's examine the different components of a backup system – software, hardware, physical location, and media handling – and consider how they contribute to reliability.

Software

Most operating systems these days come with some form of backup software, some of it more (or less) suitable for the purpose than others (see [2] for a review of the good and the bad). The best of the "stock software" lot is typically the dump/restore combination – not perfect, but they tend to do a "reasonable" job. Many sites have written home-grown wrappers around the stock commands, with varying degrees of complexity. You can try to deal with labelled tapes, tape switching, unattended execution, tape cataloging, etc., but if your disks are small enough (or your tapes big enough) your backup script can be very simple. I once set up for a small, isolated UNIX machine a backup "system" that involved a clerk putting the day's tape in the tape drive, signing on as "backup" (which ran a backup script as its shell), dropping the previous day's tape in the campus mail to an off-site location, and going home, leaving the backup running – dead simple, but appropriate for the situation. At the other extreme of homegrown software, I've seen systems that track tape numbers and maintain a flat-file index of which filesystems from which hosts are on which tapes.

But unless your needs are very simple, I'd recommend against rolling your own and reinventing the wheel. If cost is a concern, I would recommend that you investigate the Amanda Backup Manager, available from the University of Maryland [3]. It's built around standard utilities (such as dump/restore, GNU tar, etc.), does labelled tapes, has some support for jukeboxes and tape changers, maintains a database, and works on a wide variety of systems and across the network. It is very good software and definitely worth a look. There are other freely available backup systems that you might want to have a look at, including the "Ohio State" backup software[4].

And as for commercial software, there is a wide variety to choose from – pick up any industry magazine and check the ads. The commercial products typically provide a GUI management interface, online indexing of files, user-initiated restores, support for more types of hardware, etc. The commercial software seems to vary in features, different OSs that are supported, hardware support, security, polish, and price. Most have reasonable scheduling capabilities, some use "standard" tape formats, and some use proprietary formats. Have a look and see what fits your needs best.

And while we're talking about software, let's talk about "live" vs. "offline" backups. By that I mean do you run your backups against a machine running in normal "time-sharing" mode, or do you shut down to "single-user" mode and kill off unneeded processes to ensure that nothing changes on a filesystem while you're backing it up? The answer is, it depends (yes, I know that's a cop out). If you can identify a time of day when your systems are likely to be lightly loaded, and your backup software can handle filesystem changes in a "reasonable" fashion, you'll likely want to do live backups and leave your systems up and running while they're being backed up[5].
This would be a good place to talk about special-purpose software for database backups, except that that’s a bigger, more complicated subject than I want to cover here. And I could also talk about special filesystem support for backup ease (snapshots, locking, etc.), but I won’t.

As for software reliability, it’s pretty much the automation, the ease of use, and the robustness of the software that you choose that are the relevant issues. You’ll want to automate as much as possible, but with something as important as backups, you’ll want to have positive confirmation that your backups are running properly (by reviewing logs, mail messages, etc., on a daily basis).

Hardware
The hardware that you choose for doing your backups affects your ease of use, media cost and reliability, and ease of replacement in case of failure.

Most backups are made to some form of magnetic tape. In the old days, we relied on good old nine track reel-to-reel tape, but there aren’t many people investing in that technology these days. In the UNIX environment, two of the most common tape formats are 8mm (popularized by Exabyte) and DLT, with a variety of other types (DAT, AIT, etc.) also in use. A comparison of tape formats is more than I want to get into here, but I will mention a few things for you to consider when choosing a tape format:

- Reliability and intended use. Was the tape format developed primarily for data use, or is it a format that was originally developed for audio or video use? How will it stand up to your expected usage?
- Capacity. Do you need a high-capacity format, or is a lower capacity (e.g., only a few gigabytes) enough to meet your needs?
- Media cost and availability. How much does each tape cost, and how easy are they to obtain? Is it important to be able to go down to the local consumer electronics store when you run out of tapes some evening?
- Drive durability and availability. How well will the drives stand up to your expected duty cycle? Will you wear out the drives on a regular basis? Will you need quick replacements? And what’s the warranty and service contract like?
- Compatibility. How does the format fit in with what’s already in use in your organization? If everyone else is using a particular format, you might be better off to conform, so that you can borrow media, drives, and expertise if and when you need them.

The next consideration is aggregation and automation. By this I mean the choice between single tape drives and tape jukeboxes or autochangers (of small or large capacity). This depends a lot on how much data you need to back up, how long you want to keep it available, how often you expect to need to restore data, and how you expect your needs to change in the future. In many cases, the extra cost of a small jukebox (ten tapes or so) will be well worth it in terms of ease of use. And closely related to aggregation and automation is the question of how many drives you need (or want) to have. How much capacity do you need? Do you need multiple drives running in parallel in order to get all your backups done in the time available? Do you need to be able to duplicate your tapes to guard against media failure or to take off-site? Do you want to be able to do restores on one drive while doing backups on the other? And finally, if you do choose a jukebox, will the tape drives still be usable when the tape-changing mechanism breaks (as it almost certainly will sometime)?

When we were looking for a new backup system in 1996, we ended up choosing a DLT jukebox with two drives and room for 250 tapes, with the ability to expand both the
Most people learn a “backup lesson” at some point. I’m just lucky that mine was more abstract than most.

number of drives and the number of tapes. (We actually bought two of them.) This may seem like a giant system, but it’s actually only a mid-size in the world of backups, and in review, it seems to have been a good choice for our situation.

And finally, you’ll need a machine to run your tape drives. Choose a machine that fits with your other machines, and consider dedicating it to the task. It may seem a waste for a nice machine to be sitting idle all day, just to wake up and write a few tapes in the middle of the night; but for something as important as backups, it’s often nice to have a secure, limited-access machine that you can dedicate to the process.

Those of you who have been paying attention will have noticed that I didn’t mention other media, such as magnetic or optical disk. I’ll contend that those alternatives are appropriate for backups in only very special situations and that you’ll already know if they are something that you should consider.

Physical Location
Where are you going to locate your backup server? Does it need to be physically close to your desk, near your servers, in a nice locked room with fire-suppression gear? Do you need easy access to it to swap out tapes? What does your network look like? Do you have adequate bandwidth for your backups in more than one place?

I mention these questions to get you thinking about the physical and network security of your backups and backup system. Remember that a backup system makes a nice attack target because it will contain all your data. And what happens when you have a fire in your machine room and all your servers, including the backup server (and the tapes), melt? And don’t forget about a nice UPS for your backup system. You may not be able to do any backups during a power failure (because your other machines or networks might be unavailable), but at least your backup server won’t get corrupted by a sudden power outage.

When we bought those jukeboxes in 1996, we were able to put one in a building across campus that didn’t already contain any machines of interest. We dedicated a pair of fibers to a fast Ethernet connection, built a small air conditioned room, installed an intruder alarm, and locked the backup server and jukebox in there. That way, we ended up with off-site backups without having to remember to move tapes about.

Media Handling
The main considerations for media handling are how to get your tapes off-site and how to get your duplicated/Cloned tapes into a location different than the originals. Many people overlook the need to get their backups physically away from the original disks. A fire, flood, or fire axe-wielding computer hater could put you out of business.

I’ll make my point with a short story, wherein I learned the necessity of off-site back-ups. I used to do some programming for a university professor on a PC in his office as part of a major, multiyear, externally funded research project. I came in one day, and the IBM PC AT (it was a long time ago) was gone, along with every 5 1/4” diskette in the office, including the backups. Fortunately, we had another set of backup diskettes that were fairly recent at the professor’s home. Without those, we would have been in major trouble. (Most people learn a “backup lesson” at some point. I’m just lucky that mine was more abstract than most.)
Testing and Recovery Practice

The classic UNIX backup horror story involves multiple filesystems being backed up to a single tape, a hapless system administrator who accidentally specified the rewind instead of the nonrewind tape device, and a company president who just accidentally deleted a very important file.

There are two main reasons for testing your backup system. The first is to ensure that you’re actually creating good backups, that you can restore from, and that you’re backing up the files and directories that you actually intended to back up. Write a script to step through your tapes to check the dump headers on each file and generate a report. Pick some files at random from various machines, and make sure that you can find them on your backups.

The second reason for testing and practice is to ensure that, when the emergency comes, you know what to do and how to do it. When the root disk on your main central server gives up the ghost, make sure that you can rebuild it from your backups. This is a convenient place to note that sometimes commercial products that generate backups in “native” formats are a real blessing. Many operating systems let you easily restore a dump file onto a new blank disk. But if you’re using backup software with a proprietary tape format, you may have to do a complete OS installation, install the backup software, and only then start doing the actual restore. (I’ll point out that it’s convenient to be able to attach a new disk to some other running machine, do the restore there, and then install the new disk in the broken machine.)

Next Time

Next time I plan to talk a little bit about disaster recovery and the kinds of things that you will need to consider when thinking about what to do if a disaster ever strikes your organization.

Notes


[5] You may wish to consider a full, offline backup before you do OS upgrades or hardware changes. If something goes wrong, it can be very comforting to know that you’ve got a nice safe backup nearby.
Toolman Features Steve Kinzler

Highlighted in this article: webrowse, a tool for viewing the world (or text) through the eyes of a browser.

Open for Business
When I first visited Steve Kinzler’s Web page, it was like I had stumbled into an old-style hardware store. Tools were hanging on the walls and lying on shelves, some of them elaborate and exotic, others simple and mundane. He had responded to one of my solicitations for tools and had invited me to visit his shop, enticing me with the prospect of custom-made tools of unique character.

And, well, Toolman can carry on with this figurative language for only so long. Suffice it to say that Steve is a system administrator after my own heart, who looks for creative solutions to simplify and expedite the common, repetitive, computer-related tasks that beset him and his users, and who authors software tools as a means toward that end.

Text 2 Browser
One of the tools that Steve highlighted in our correspondence is a program called webrowse, which can be used as a quick interface to a Web browser (hence the name) on a UNIX system.

webrowse is a handy tool for us command-line types living in a point-and-click world. With a browser running somewhere on your workstation (even iconified or in another virtual desktop), point webrowse at an HTML document (a file or STDIN), and it will bring it up in the browser; point it at some plain text, and it can first mark up the text, adding appropriate hyper-text links on the fly. Now some of this might sound like something you could do with a few simple aliases, but various aspects of this are not so easily handled.

webrowse can currently interface to both the Netscape (default) and Mosaic browsers, selectable via command-line or environment, and will issue appropriate commands to activate an already running browser. (Both of these browsers have the “remote control” features that webrowse exploits.) With the -m (markup) option, it will HTML-ize the input by adding standard HTML header and body tags and by scanning the text for anything that looks like an address or a URL and adding an appropriate link. So, for instance, an email address will be marked up with a mailto: link. Other possible markups include links for http:, ftp:, file:, and news: webrowse employs sophisticated pattern matching as a basis for its heuristic approach to these transformations.

webrowse is also handy as a filter (with -o) such that the converted text output can be directed to a file or piped on to some other process.

webrowse Examples
As a simple example, let’s say you have an email message about virus hoaxes that contains some URLs for Web pages on this topic. You can save the message to a file named “virus-hoax”, then type webrowse -m virus-hoax. The text of the message will be marked up with HTML tags, including the URLs, and will then pop up in your browser window, where you can follow the links easily. A more efficient method would be to map some function of your mail reader to the command webrowse -m or just pipe the message to this command, reducing steps and avoiding the need for the temporary file holding the message (and you know how Toolman despises temporary files!).

by Daniel E. Singer
Dan has been doing a mix of programming and system administration since 1983. He is currently a systems administrator in the Duke University Department of Computer Science in Durham, North Carolina, USA.

<des@cs.duke.edu>
As another example, Steve uses the following key mapping with the \m newsreader:

```
map both I {
  save-full ";webread -mw"
}
```

This allows him, by hitting \i, to view the selected articles in a new browser window, with all the URLs, email addresses, etc., converted to links.

He also defines some macros in his .exrc file for use with the vi editor. For example:

```
map "V"Iwb :w !webread -m"M
map "V"Iww :!webread %"M
```

The first (wb) will bring up the text currently being edited in the browser with markup added. The second (ww) will tell the browser to load the current file. For each of these, type TAB and the two letters to invoke the macro.

**webread** has a plethora of command line options and environment variables for fine tuning and customizing its operation, making it handy to embed in other scripts as well as use on its own. The `-h` (help) option and the \man page will shed some light on these.

**Other Aisles**

Steve's shop, er, Web page includes many other tools addressing various aspects of system administration and general UNIX usage. Here's a quick survey of a few that might warrant an evaluation:

- **Web:**
  - **ClipControl:** this Java applet is an AudioClip controller for flexibly embedding audio files in Web pages.

- **Web administration:**
  - **ftw:** file tree walker, for Web document tree checking. Checks validity of symbolic links, especially if the server's running with `FollowSymLinks` set.
  - **starthttpd:** start, restart, or kill an HTTP daemon, as needed. Helps to keep a server up near 100%.
  - **rollogs:** rollover NCSA-style httpd log files, works with `starthttpd`. Flexible roll-over of Web logs at various resolutions.

- **Systems administration:**
  - **dumpdates:** produce readable and organized listing of dump dates for mounted filesystems.
  - **rdistsum:** produce readable summary of `rdist` output, highlighting errors.

- **General use:**
  - **push** and **pop:** conveniently and safely push/pop files into/out of a subdirectory.
  - **rename:** move or copy files and directories based on a `sed` or `perl` expression.
  - **vgrep:** edit all files containing the given regular expression, such as for multi-file software development.
  - **wh:** list all instances of given files in a search path. When `which` just isn't enough . . .
  - **width:** determine the printing widths of input lines, find the longest line in a file, etc.
Many other public domain tools for converting between various formats and HTML are available.

z: convenient, safe front-end for (un)tarring and (un)compressing, with intuitive use of subdirectories. z <something> usually does the right thing. Good for naive users.

About the Shop Proprietor
Steve hung his hat for quite some time at Indiana University, Bloomington, where he completed his M.S. in computer science, taught, worked on many projects, and performed Web and UNIX systems administration. He is creator and maintainer of the Picons database (<http://www.cs.indiana.edu/picons/ftp/>) and the Internet Oracle (a.k.a. the USENET Oracle) (<http://www.pcnet.com/~steno/oracle/>), and is a longtime member of USENIX. His other accomplishments are far too numerous to list here. He currently lives in Ann Arbor, Michigan, and is working for the Health Management Research Center at the University of Michigan.

Thanks, Steve, for making your materials available.

More Browsing
I surfed the Web a bit to see if other tools similar to webrowse were available. I found one called txt2html by Seth Golub that has some interesting features. It's quite versatile in its ability to add markup and allows you to define a private "dictionary" of conversion rules. It is strictly a filter, and lacks the ability to automatically interact with a browser. (But you can do txt2html < foo.txt | webrowse -s). txt2html can be found at <http://www.cs.wustl.edu/~seth/txt2html/>.

Many other public domain tools for converting between various formats and HTML are available. A good starting point for a search is <http://www.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/HTML_Converters/> . (Try looking that one up in your Funk and Wagnallst!)

By the way, on the topic of conversions to/from HTML, I've written a script called index2html that can be used in conjunction with the check program (June 1997) to create interconnecting HTML-ized INDEX files in a directory hierarchy. index2html is still in its adolescence; comments are welcome. It can be found at <ftp://ftp.cs.duke.edu/pub/des/scripts/>.

As a final example, here's how I've used webrowse and index2html together. Somewhere in our extended filesystem, we have a directory hierarchy of documentation rooted at /home/1ab/doc/. Each directory has an INDEX file, and index2html is used to generate the INDEX.html files. The following script, called 1abdoc, easily brings up a browser displaying the top level of this mini-web of documentation.
#!/bin/sh
#
# @(#) labdoc: bring up a browser on the /home/lab/doc/ hierarchy
#
# this script uses the script `webrowse', which will bring up
# the document in an already running browser; if that fails, a
# new browser is started;
# `WEB_BROWSER' is one of the environment variables recognized by
# `webrowse', and is used here for consistency;
#
# 1/98, D.Singer

DOC="/home/lab/doc/INDEX.html"
WEB="/home/lab/bin/webrowse"
DFLT_BROWSER="netscape"

WEB $DOC 2>&- | |

{ $(WEB_BROWSER:-$DFLT_BROWSER) $DOC & }

exit

Closing time

What we've seen here, via examples from Steve's Web page and beyond, is the tool
approach in action: an approach that is very natural to the UNIX environment. A well-designed tool can make your life easier and can often be utilized as a component of
other tools, as in the labdoc example above. And the design process can make life more
interesting (to us command-line types, anyway). Sh, Perl, Tcl, . . . , these are powerful,
high level languages (I've even seen an example of Bourne shell used as a formal object-oriented language [1]), and they're relatively easy to use. Need a tool? Write one! (Or
cop one out on the net.)

Are there any tool topics that you would like to see covered? Be sure to let me know if
you have any suggestions for future articles.

Note

One ISP’s Response to the Problem of Spam

Introduction

As the Internet has become less of a research-oriented collection of computer-oriented acquaintances and more of a multinational business-driven communications medium, it has had to deal with problems that simply didn’t exist before. One of these is “spam,” the Internet equivalent of bulk mailings, junk faxes, and unsolicited telemarketing phone calls. This article details what we have employed in our effort to stop spam, with the obvious hope that the techniques used and the lessons learned will be of use to others.

Background

In 1994, two Arizona-based immigration attorneys, Canter and Segal, advertised their services by sending a message to each of the several thousand USENET newsgroups, whether their message was appropriate for the discussions taking place there or not. This unusual behavior earned them public scorn, and after they continued this practice, they were kicked off of one Internet service provider (ISP) after another. These days you can’t enter a newsgroup without seeing such messages, typically called “spam,” and in some newsgroups (especially the sexually oriented ones), they make up the majority of the traffic.

These sorts of tactics have not been reserved to USENET news. More recently, individuals have taken to harvesting email addresses from Web sites they maintain, open mailing lists, and USENET newsgroups so they might send unsolicited advertisements for various products and services. Officially known as UCE (Unsolicited Commercial Email) or UBE (Unsolicited Bulk Email), these practices have also been lumped under the general category of “spam,” the definition of which has now been expanded to include essentially “all electronic garbage messages.”

One of the more significant problems with spam is, unlike telemarketing or bulk postal mail, the sender pays very little of the cost of transporting the message. The spammer simply gives a mail host (often an ISP, due to its excellent connectivity, high-volume capacity, and a general difficulty keeping track of the huge amount of mail and news that passes through its system) a list of targets with a single message to send. The senders incur very little cost per message – essentially only their time and the cost to set up an account with an ISP. The host that relays the mail pays for the bulk of the transmission in bandwidth, service degradation, and cost of responding to the ensuing complaints. The target site also pays in loss of bandwidth, disk usage, connection costs, overflowing mailboxes, etc.

Of course, the cost that most people complain about is the expenditure of time and effort to sort, read, and delete the unwelcome communique. This can be especially painful when paying for access by the minute or by the byte. Most perceive the situation as unfair and feel that the costs of sending such messages should be paid by the initiator, not by the systems that are being abused or by the recipients.

Who We Are

EarthLink is one of the largest ISPs, serving approximately 450,000 customers, handling about two million pieces of email per day. By mid 1997, EarthLink was targeted by a large number of spammers, and the sheer volume of spam going through our networks
was starting to have a significant impact on the performance of our services. At this point, we started taking legal action against the spammers and implemented the technical solutions that we present in this article. Our “zero tolerance” policy has the full backing of our upper management, and we go to great effort to ensure its implementation.

We are often asked if we claim to be so antispam, why don’t we simply throw a switch and stop it from going through EarthLink’s servers altogether? Unfortunately, there is no total solution. Although we have deployed a great many human and computing resources along with a large variety of technical and social tools, the spammers (new and old) keep misusing our resources almost as fast as we can stop them. The sheer amount of data flowing through our system defies implementation of a simple and all-inclusive mechanism by which to stop all network abuse.

We believe that the decision on what Internet traffic one wants to see or not see should be made by the end-user, if at all possible. And if one makes the decision to see or not see data coming from an individual source, one should be able to effect that decision. We also believe that anyone should have the right to refuse to provide service to those who use resources without paying for them and reserve the right to refuse to do business with anyone, especially those who consume a disproportionate amount of server, network, and human resources. People who use other people’s resources to deliver their messages without express consent are, in essence, stealing. Those who deliver a message pretending to be someone they are not are, in essence, committing fraud. We do not support anyone’s claim to the right to commit acts of this nature.

Caught in the Middle
At various times, the privilege of being the prime source of spam has made the rounds through the ISP community. AOL, Netcom, MCI, UUNet, EarthLink, Compuserve, the various regional Bell companies, and others have all held this distinction at one time or another. Not surprisingly, this title usually befalls an ISP during a time of extreme growth or other significant set of events in the history of the organization. ISPs are then forced to take their most extreme antispam measures when they have the fewest resources available. We believe that almost every entity that has been through this will recommend that organizations tighten up their systems and develop policies and procedures to deal with these situations before they arise. If an organization fails to do this when resources are available, it may be forced to face this problem when it is least able to.

Compounding the problem are the unscrupulous practices by many of the spam purveyors. Internet discussion groups are replete with stories of forged return addresses (so the advertising targets cannot complain to the true sources), hijacked servers (the spammers may use the computing resources of others to distribute their messages without permission), fraudulent identity claims (so that it is more difficult for filtering software to determine if the message comes from a source of spam), and procedures for removal of one’s email address from these mailing lists that often do not work. It is no wonder that many have declared outright war on spam; consequently, spammers have had to resort to even more aggressive hit-and-run tactics to get their messages through.

Caught in the middle are the ISPs. The subscribers to these services generally don’t want to see these messages, but if the ISP tries to filter these connections, it is accused of censorship. These organizations have the largest and most powerful email systems in the world, which their subscribers insist be more accepting than corporate servers (which can be restricted by policy and/or firewalled off), so they are natural targets for
The story is analogous to security problems in general – the solutions are widely known, but apathy, the easy cushion of ignorance, the pain of change or implementation, the lack of auditing/verifying tools, or all of the above are preventing people from doing anything about it.

relaying by the spammers. Also, ISPs provide very inexpensive and convenient access as a jumping off point for the spammers to gain access to the Internet. These problems are not all easily solved, and it doesn’t help, nor is it coincidental, that the tremendous increase in demand for Internet-capable professionals has been coincident with the time of ISPs’ most rapid growth. Although this does not excuse an ISP that has ignored these issues, it is improper to believe that you can completely judge another situation without understanding the details of it.

The Current State of the Internet
On January 30, 1998, we decided to use a modified version of SATAN to conduct a quick-and-dirty technical survey of all the ISPs listed in CNET’s “Ultimate Guide to Internet Service Providers” to get an idea of how many ISPs allowed unrestricted mail relaying. We decided to examine ISPs both because we are in the ISP business ourselves and most of the mail delivered on the Internet today is by ISPs. The results were staggering, quite simply, and go a long way to explain much of the reason why spam is such a problem on the Internet today:

<table>
<thead>
<tr>
<th>number of ISPs checked:</th>
<th>597</th>
</tr>
</thead>
<tbody>
<tr>
<td>number allowing unrestricted mail relaying:</td>
<td>320</td>
</tr>
<tr>
<td>% allowing unrestricted mail relaying:</td>
<td>53.6%</td>
</tr>
</tbody>
</table>

See the appendix for more on the details of the survey and the methodology used.

Although the final percentage of ISPs having open relays is only an approximate value, it’s easy to see that the Internet is indeed a spammer’s heaven. Even if a conscientious ISP turns off mail relaying or kicks a spammer off its systems, the miscreant can easily choose a different home or target to abuse. The story is analogous to security problems in general – the solutions are widely known, but apathy, the easy cushion of ignorance, the pain of change or implementation, the lack of auditing/verifying tools, or all of the above are preventing people from doing anything about it.

At the USENIX LISA 11 conference in San Diego in October of 1997, we held a BoF (Birds of a Feather) session on the problem of spam. It became apparent to us that the Internet community was extremely hungry for any advice on practical methods to reduce spam. In this article, we hope to provide practical information on how to help protect one’s systems and to provide insight into how a site (be it an ISP or otherwise) might structure its response to these sorts of problems. Unfortunately there appears to be no single solution to this problem.

Technical Anti-Spam Methods
We can roughly divide our antispam efforts into technical and social methods.

Although both have proved effective at reducing spam, the spammers are sometimes wily and always tenacious and have been very adaptive in combatting our efforts.

Technical efforts to stop spam are, of course, favored by us (being longtime Internet geeks). Realtime monitoring is fascinating, but very difficult, if for no other reason than the sheer volume of data flowing through our networks. So we’ve tried to focus on proactive methods whenever possible.

UNIX Mail Relay Filtering

Ever since email was sent via the Internet, people have generally configured their machines to accept and attempt to deliver any and all email whatsoever. If their host was not the final destination, it would be dutifully forwarded to the appropriate machine. Indeed, UUCP and early Internet mail would never have worked if this were not true.
This was part of the general philosophy of the early Internet: be a good neighbor; be generous in what one receives and restrictive in what one sends. Thus, if I sent an email message to <zen@trouble.org> from <npc@am.org>, but sent it via the SMTP capable machine <mail.earthlink.net>, this server would have dutifully tried to deliver it to the <trouble.org> mail server for me.

This has been subverted by the spammers to (1) make somebody else do the hard part of delivering mail messages, (2) get around an administrative block of this spammer’s organization, and/or (3) mask their culpability in this act. For example, let’s assume I’m a spammer dialed up to my ISP, and I’m currently logged on to their service at <diaplus666.faus.isp.net>. Now, I have a list of email addresses, maybe many thousands of them, to which I want to send an ad. I connect to <mail.good.isp.net>, claiming to be <niceguy@innocentcompany.com>. I then give a list of addresses to send my ad to, and the mail server will dutifully try to send the mail.

This open relaying policy is a friendly thing, in the best tradition of the Internet. On those rare occasions when an email message might get misrouted, machines will try to straighten everything out in a spirit of openness and cooperation. Before the rampant commercialization of the Internet, nobody thought twice about relaying mail for other sites, especially if they spanned networks. In fact, there were several sites that openly offered to do this as a public service. Unfortunately, this has been so badly abused by the spammers that the practice is on its way to being a distant memory on the Internet today. Here is how you can set up a system running the sendmail SMTP agent to prohibit unauthorized mail relaying for trivial and more complex cases.

**Simple Case**

The easiest way to prevent mail relaying is to simply disallow it altogether. The vast majority of hosts on the Internet can be set up this way. In fact, if the machine in question does not provide remote mail access (typically via the POP or IMAP protocols) or is not a central mail hub, this is undoubtedly the way the machine should be set up.

In order to block relaying in this manner, you need to be running the freely distributable version of sendmail, version 8.8 or higher. If you are not running at least this version, an upgrade is in order in any case because of the security problems associated with earlier versions.

In the `sendmail.cf` file, you simply need to add the following lines, stolen from the antispam rules at the sendmail Web site:

```plaintext
Scheck_rctp
# anything terminating locally is ok
R< $+ @ $=w > $@ OK

# anything originating locally is ok
R$* $: $(dequote "" $&{client_name} $)
R$=w$@ OK
R$@ $@ OK

# anything else is bogus
R$* $ERROR $: "550 Relaying Denied"
```

These lines can be placed anywhere in the `sendmail.cf` file as long as they’re not in the middle of another rule set. We like to put ours at the beginning of the file just before the “w” macro is defined.

You do not need to do anything more than add these lines and restart the sendmail daemon for the rules to take effect. These rules operate only on the envelope of the mail message, not the header, so that sendmail can’t be fooled by forged headers. If the send-
We urge system administrators to consider using the prospect of mail relaying as an impetus to rearchitect their mail systems, where appropriate.

Mail daemon receives email that either is not bound for the machine in question (that is, the machine in the "RCPT TO:" field of the envelope does not match the list of machines in the "W" macro of sendmail) or is not sent by itself, it rejects the connection with error 550 and the message "Relaying Denied." This is the way we recommend all machines that aren't mail hubs (e.g., desktop machines that need to run sendmail in daemon mode) be set up.

One final thing to note is that it's a bad idea for most hosts to run sendmail in daemon mode at all. Despite the fact that UNIX workstations come out of the box with sendmail installed (and almost always with relaying enabled), it is rarely necessary to run sendmail on more than a small fraction of computers on a given network. We urge system administrators to consider using the prospect of mail relaying as an impetus to rearchitect their mail systems, where appropriate. If your systems have been abused in this manner (if not by spam, it might be enough to remember that sendmail is one of the prime ways that intruders break into computers these days), you'll probably find this to be a relatively easy sell to your organization and/or management.

**Advanced Case**

What if a mail machine does act as a POP or IMAP server? In this case, there very well may be legitimate computers that need to use this machine to relay mail. You can specify a class – we use "W" – to be the hostnames of allowed relayers. If DNS isn't set up as well as we like, we additionally specify a class, "C" of all the Class C networks allowed to relay through us. The whole section to be added to the sendmail.cf looks like:

```bash
Scheck_rcpt
# anything terminating locally is ok
R< $- @ $w > $@ OK

# anything originating locally is ok
R$* $: $(dequote ""$&{client_name} $)
R$W $@ OK
R$+ . $=W $@ OK

# IP address ranges
R$* $: $(dequote ""$&{client_addr} $)
R$C . $= $@ OK
R@$ $@ OK

# anything else is bogus
R$* $#error $: '550 Relaying Denied'
```

For this to work, we need to add the following definitions, also in the sendmail.cf file:

```bash
# file containing domains which are allowed to relay through us.
FN-o /etc/mail/sendmail.cW

# file containing legitimate client relayers by Class C prefix.
FC-o /etc/mail/sendmail.cC
```

The file `/etc/mail/sendmail.cW` might contain something like:

```
earthlink.net
trouble.org
```

and the file `/etc/mail/sendmail.cC` might look like:

```
208.197.253
207.217.91
207.217.118
```

That's all there is to it. Now mail will be rejected by this machine unless one of the following conditions holds:
The destination host of the mail is in the contents of the “w” macro.

The source host is in one of the DNS domains that are acceptable relays (from the file /etc/mail/sendmail.cf). As an example, the machine <trouble.trouble.org> would be allowed to relay; its domain, <trouble.org> is in the sendmail.cf file.

The source host is in one of the Class Cs that are acceptable relays (from the file /etc/mail/sendmail.cc). As an example, the IP address 208.197.253.128, whose network is in the sendmail.cc file, would be allowed to relay.

Mail is being sent from a process on this machine.

For those who aren't as familiar with the sendmail.cf file syntax, machines listed on a line that begins with Dw, Cw, or in a file called sendmail.cw make up the complete list of machines and domains for which the machine in question stores mail.

Of course, you can play with the rules, changing the Class C networks to Class Bs, removing the domain checking rules, or whatever is appropriate.

Testing these Rule Sets

Of course, you don't want to simply put these changes in and hope they work. They need to be tested. First, you should get an account on a machine from which relaying should not be allowed (not the machine that sendmail is running on!). For example, if the machine with the new relay rules is named <death.trouble.org>, you should Telnet to port 25 of this host from a disallowed host and verify that regular mail works but relaying doesn't by doing the following (the typed commands are in bold text):

```
fish.com % telnet death.trouble.org 25
Trying 208.197.253.134...
Connected to death.trouble.org.
Escape character is '^]'.
mail from: npc@acm.org
250 npc@acm.org... Sender ok
rcpt to: <npc@death.trouble.org>
250 <npc@death.trouble.org>... Recipient ok
rcpt to: <npc@acm.org>
550 <npc@acm.org>... Relaying Denied
quit
```

Mail is accepted for the local machine and denied for destinations not in the sendmail “w” macro.

Now we test from a machine that should be allowed to relay.

```
trouble.trouble.org % telnet death.trouble.org 25
Trying 208.197.253.134...
Connected to death.trouble.org.
Escape character is '^]'.
mail from: npc@death.trouble.org
250 npc@death.trouble.org... Sender ok
rcpt to: <npc@death.trouble.org>
250 <npc@death.trouble.org>... Recipient ok
rcpt to: <npc@acm.org>
250 <npc@acm.org>... Recipient ok
```

In this case, relaying was not denied whether the mail was to be delivered locally or not. These rules work and are probably safe to implement. As always, when making changes to the sendmail.cf you need to restart the sendmail daemon for them to take effect.
USENET news spam has been around longer than its email cousin, but it turns out to be fairly easy to implement a technical solution that greatly curtails it without serious side effects.

There are two things to note. First, the angle brackets when typing in the “rcpt to” line are mandatory. If these are omitted, you will always get “relaying denied.” On machines without the Scheck_rcept ruleset present, you will get “Sender ok” if they are omitted, but they are required by the SMTP protocol. Second, what is typed as an email address in the “mail from” line is irrelevant as long as it’s a proper SMTP email address. This is never checked. Only the hostname/IP address of the sending host and the “rcpt to” line are ever checked by these relay rules.

You can do these things and more using the Scheck_relay rule set, but it’s been our experience that using this rule set is buggier, slower, and rarely necessary. Nonetheless, information on these rules and others like them can be found at the sendmail Web site or in Sendmail 2nd ed., by Bryan Costales with Eric Allman, published by O’Reilly & Associates. Both sources are highly recommended.

We also created a SATAN testing module that can be run on individual hosts or large networks. See “Auditing Tools” for more information on this.

News Backoff Algorithm

USENET news spam has been around longer than its email cousin, but it turns out to be fairly easy to implement a technical solution that greatly curtails it without serious side effects. The NNRP daemon is the process on an INN-based USENET news server that receives newsreader client connections; that is, this is the process on the news server to which the news client connects. The first thing you must do is make sure that posting is restricted to those hosts that should have access to the news server. The file that restricts access is called nnrp.access and is located with the rest of INN’s configuration files. The exact location is operating system and version specific. Configuring this file is relatively simple; consult the nnrp.access(5) man page.

Additionally, what we’ve done is modify the NNRP daemon to keep track of how many posts come from a particular IP address in a period of time. If either the threshold for number of articles per unit time or the total number of articles is exceeded, the nnrp daemon goes to sleep for a few seconds. The sleep time exponentially increases with each new successful post until a maximum value is reached; of course, if the posting attempt fails, nnrp recognizes this and resets the counter after a period of time.

This algorithm has been very successful for us on our news service. We have drastically cut down the spam sent through our service without eliciting too many complaints. Indeed, we have found that if these (configurable) values are set properly, very few human posters will notice this policy change while any overly prolific automated posting program will quickly slow down to a crawl.

The backoff patches to INN’s nnrp( for INN version 1.4unoff4) are available to the public. Dave Hayes came up with the idea and wrote the patches while under contract by EarthLink Network. We expect these options to be added to the base INN distribution in the near future.

Of course, despite our best efforts and intentions, this can adversely affect some legitimate users. The first class of these is the frequent binary posters; their robot posting programs are, as far as this algorithm is concerned, indistinguishable from spammers. The second class of users that might notice this is those who use offline newsreaders. They slurp down piles of articles, read them offline, generate their responses, connect to the news server, and then send them up in one big batch. If the initial threshold is set to a number above what even an extreme news poster is likely to want to post in a single session, they won’t be affected. Even if they are backed off, it may not be a problem
for them because the postings will get through eventually, albeit slowly. If they are paying for connect time charges, though, this could be more than annoying.

The number of subscribers we have encountered who are legitimate users of the system but have been significantly affected by this change in service has been very small. For those who need to do robot posting, you could try to provide an authenticated NNRP service for them to post with. The details of such a news protocol have not been incorporated into an Internet standard, but the latest version of INN interoperates with several authenticating news clients.

**Auditing Tools**

We started by writing a simple script for parsing INN logs to assist humans in identifying spam. It examines news headers and reports on suspicious items – nonlocal email addresses, stereotypical spammy subject key words (“FREE,” “PANTIES,” and others too explicit to print in a family periodical), excessive cross-posting, a single person posting too many messages, etc. However, this is simply a reactive tool. Ideally, we want to stop the spam before it starts. Nonetheless, we believe it is impossible to stop completely, so maintaining a battery of reactive tools is necessary.

**SATAN Module for Relay and News Checking**

We also created a module for SATAN that can systematically walk through a network detecting if any hosts allow mail relaying or VRFY and EXPN queries or are running unrestricted NNTP servers that may need to be protected. This will be packaged with the next release of SATAN and will be available at <http://www.trouble.org/satan/spam.html>. It’s remarkable, even with a fine system administration staff and a conscientious technical crew, how many systems continually keep cropping up with these sorts of problems.

**Additional Logging – RADIUS Accounting**

Another problem ISPs face is identifying service abusers in realtime. If caught “in the act,” there is little room for argument as to whether they are responsible, and an immediate response can be taken. Making a mistake here is both unfair and bad for business; therefore, it is important to make this as accurate and efficient a process as possible.

Most dialup access equipment can be set up to use the RADIUS protocol to authenticate users’ access to an ISP. An extension to this protocol, RADIUS Accounting, was designed to communicate accounting information between network access gear and an accounting server and is an exceedingly valuable tool that can greatly help in identifying a resource abuser, albeit after the fact. Unfortunately, this is still a problematic solution for us, primarily due to a lack of interoperability standardization and some very poor vendor implementations of this relatively new protocol. However, with the release of RFC2159, which standardizes RADIUS Accounting, we have further hope that sites will be able to support a RADIUS Accounting service stably across a variety of dialup access platforms.

If you have a large number of dialup ports, setting up a RADIUS Accounting server can require a significant amount of planning and resources. The service must be stable, accurate, and reasonably speedy, or it isn’t going to do any good. Therefore, some thought and planning about how this service will be set up and maintained, as well as about what tools need to be written to access this information, need to be expended. If you are a smaller ISP or otherwise have a small dialup pool to maintain, the RADIUS Accounting logging code in the standard distribution can suffice, but larger services need to plan carefully for the very large volumes of data this service can generate.

**Ideally, we want to stop the spam before it starts. Nonetheless, we believe it is impossible to stop completely, so maintaining a battery of reactive tools is necessary.**
Despite our best efforts in the technical arena, we have discovered by far the most important ingredient in reducing spam is not technical in nature.

As a final warning, even with RADIUS Accounting, we (like many other ISPs) have an additional logging problem. Because we lease POPs from other ISPs (primarily UUNET) and therefore don't own all the resources involved, our people trying to identify the abuse of our systems will not always be the ones who are able to identify the account. This discontinuity makes it doubly important to have a single point of contact internal to EarthLink to manage and facilitate all communications for any network abuse.

Social Antispam Methods

Spam Cowboy

Despite our best efforts in the technical arena, we have discovered by far the most important ingredient in reducing spam is not technical in nature, but is simply having a single person who understands both the technical and legal issues involved and personally handles the whole investigative process from beginning to end. This includes (but is not limited to) watching the logs (manually or assisted by automated processes) for suspicious behavior, determining if the records indicate potential abuse, deducing the originating host, checking the specific piece of access gear or logs for the abuser's identity, and at least recording that person's identity for possible action by the ISP. The basic idea is to eliminate any intermediaries in determining who is responsible for a given infraction. Taking the appropriate action should be done as quickly as possible because even a single abuser can do a lot of harm in a relatively short period of time! In addition, rapid responses by the ISP to spamming incidents tell the attackers that it would probably be unproductive to attempt further abuses if they were to sign up again with this particular organization.

Because this is a relatively new job description, it's nearly impossible to find people who have any experience to fill the position. Every organization must either develop these resources in-house or steal them from another ISP. Not only must candidates for the position have a good level of technical competence, a well-developed sense of ethics, and a good set of social skills (interacting with individuals at other ISPs and organizations, law enforcement personnel, and customers demands this), but they also must have a very thick skin. Complaints from the general public, griping from the subscribers, and telephone calls from the abusers (which can even take the form of death threats!) are a daily occurrence. It really is a tasking job, and it is difficult for those who haven't experienced it firsthand to understand its demands.

Punishment

One controversial measure we employed was to modify the EarthLink Subscriber Acceptable Use Policy (AUP) to include a provision to charge $200 to a subscriber who commits acts of network abuse, which include spamming as we have described it here. Employing this was not without considerable controversy within EarthLink, and we had to lobby our legal department and upper management to get it passed. Fortunately, having worked closely with them throughout the process, we haven't experienced any negative legal fallout from imposing these fines.

Collecting the fines turned out to be very simple: we charge the credit card used by us for billing. From the start, we consciously tried to reduce the number of "friendly fire" accidents by focusing on the more egregious offenses and using these as examples. This way we've managed to hit back hard against the really bad offenders while sending a message to casual spammers that they should think twice before using our service for these purposes.
As with any policy change, it was vital to get the message out to our subscribers. We modified the AUP on our Web page, sent email to all of our subscribers detailing the changes, and printed an article in our bimonthly newsletter, *Blink*, which is also online. The response we received from our subscribers regarding the changes we made in this regard have been overwhelmingly positive. The problem is well known and understood, and our candid description of what we were doing about it and how it would affect our customers was very well received.

This has been a big success for us, and we heartily recommend that other ISPs consider adopting a similar measure. It does require some serious work to accomplish, but we have found it to be more than worth pursuing.

**Negative Solutions**

There are several sets of measures that folks have taken on the Internet in an attempt to deal with spam that weren’t mentioned here because we do not like them. Some so-called solutions are, in our opinion, not solutions at all, for they advocate an eye for an eye (or worse) philosophy. We feel that in some cases these “solutions” are at least as worrisome as the spamming problem they’re attempting to solve, and we do not recommend that they be adopted.

**Terrorism**

Foremost among these are what can only be described as terrorist attacks: Ping-of-Death, mailbombing, smurfing, hacking, and other denials of service or outright attacks against both the spam purveyors and the unwilling accessories to their offenses. These attacks are worse than the spammers, for although they are typically out for monetary gains, terrorists have real malice behind their actions with an intent to injure. In addition, these efforts can often have far-reaching and unintended consequences, not only to their target, but also to innocent victims along the path of destruction.

**Black-Hole Routing**

Another so-called solution that some folks have adopted to combat the spammers is to fail to route their networks; at the last LISA, one such group claimed to have a set of participants that could eliminate a target’s capability to see about 20% of the Internet by blacklisting it. Although we believe that a terminal or endpoint network certainly has the right not to accept traffic from places it does not wish to communicate with, potential abuses have made this a practice we cannot support.

First, transit networks should not do this, only endpoint networks. As an ISP, we should not prevent the folks to whom we provide service from being able to contact anyone that they choose on the Internet. Under no circumstances should we censor their access without their express consent. If they ask us to filter, that is an entirely different matter and acceptable.

Second, on more than one occasion, legitimate users have been cut off from a significant portion of the Internet accidentally, despite their innocence of any form of network abuse. We cannot, in good conscience, support a system where this is such a strong probability.

Third, this solution, as it is currently implemented, bestows a great deal of power to an individual, so a potential for abuse is there. Even though we don’t suspect that any unethical activity is likely, the mere possibility of this is distressing.

In addition, the misconduct of these individuals can make the spammers appear to be victims, rather than the network abusers that we believe they are.
We do not support these sorts of activities in any way, shape, or form, implore the employers of these methods to desist, and call for other legitimate organizations to decry these methods as well.

**Future Work**
There’s a lot more still going on that will or may build on the efforts we have outlined here. Unfortunately, they aren’t all positive or constructive efforts, in our opinion.

**Sendmail’s No-Relaying Default**
Starting with sendmail 8.9, sendmail will have mail relaying off by default. This should cut down the amount of open relays by a considerable margin, because it is still by far the most popular mail delivery agent on the Internet.

**SMTP Backoff**
Since our USENET news backoff solution was so wildly successful, we’ve turned our attention to doing this for SMTP as well. We’re currently talking with Eric Allman with the hope that he will add these capabilities to sendmail. We would like to see the mail local recipients stream through unaffected while outbound mail being relayed through the mail system is subject to the same basic kinds of backoff procedures we use for news. There’s no completion date on this, but you might want to start looking for it sometime in late 1998.

**Realtime Monitoring**
We have the unenviable (from a security perspective) situation of having a large amount of network traffic and bandwidth that will only grow larger. Trying to monitor 50MB a second of email in realtime is a difficult task at best; we have yet to find something that can keep up with this volume of traffic. However, with the recent release of Network Flight Recorder, a programmable, high-speed, network-monitoring tool, we are hoping to put more significant effort into solving this problem. Having a tool that could warn us of network abuses as they occur could help us greatly mitigate our current dilemma. It remains to be seen whether this or any network-monitoring tool can keep up with present and future load.

**IP Caller ID**
We are envisioning a system whereby a unique identifier is handed out to a computer with a dynamic IP address when it signs on. This information can be used to grant or deny access to individual client machines within a single piece of dialup access gear. In this way, multiple ISPs could share a common dialup access provider without making themselves vulnerable to network abuse by the other ISP subscribers using IP-based authentication alone. This idea is hardly even in its infancy, but it is a technical possibility that might be worth pursuing.

**ISP Version of NCTDE**
Starting in December 1997, the telephone long distance phone companies (IXCs) put into full service a blind database maintained by an external entity for the purpose of coordinating information on households that are bad credit risks, that is, jump from one long distance provider to another without paying their bills. This database is known as NCTDE (National Consumer Telecommunications Data Exchange). Because of the way this database is structured, the phone companies have obtained an antitrust exemption from the Department of Justice. A nearly identical system, called NTDE
(National Telecommunications Data Exchange), has been in service for over two years to track businesses in this matter.

No research or work that we're aware of has been done on this yet, but it seems reasonable that a similar service might be implemented in the ISP world and that this service might be expandable to track spammers.

**Alternate Mail Delivery Systems**

Although there have always been alternatives to sendmail, there has never been a serious challenge to its supremacy as the UNIX mailer of choice on the Internet. However, two mailers have stirred up quite a bit of interest and popularity: Qmail by Dan Bernstein and the upcoming VMailer by Wietse Venema. Both have various antispam features and, of course, have mail relexing off by default.

**Resource Sharing Among ISPs**

Resources can include realtime information, as well as personnel, hardware, and software. Rapid and easy communication among ISPs on resource abuse may have a great deal of promise in reducing the overall impact of the spammers on the Internet, although there are significant technical and legal barriers to making this happen. However, we hope groups like IOPS will help establish a dialog on how the ISP industry as a whole can cooperate to reduce the spam problem.

**Laws**

For better or for worse, it is primarily through legislation that governments have such an enormous impact on how the Internet functions. Opinion is currently split between those who believe that a legal approach would be a productive way to attack the spam problem and those who believe that government intervention is more to be feared than invited. We believe that both of these viewpoints are reasonable and are ourselves split on this issue.

Until now, the US Government has mostly let the Internet grow and evolve in a fairly unfettered state. This, combined with the overwhelming success of the Internet, has some fearing that if the government does intervene on the issue of spam, it will be an invitation for even more legislation on other issues that will have undesired consequences.

The often cited junk fax law (47 USC 227) has had a powerful and beneficial effect on curbing this nuisance in the fax world, and it's easy to understand why many folks believe that extending it to cover UCE and USENET news spam would be very beneficial. If an antispam law would have an analogous impact to the junk fax legislation, it would be hard for anyone who opposes spam, no matter how anarchistic he or she might be, not to concede that such legislation is a good thing. Indeed, even if some small interference by government in other Internet areas were a consequence, on balance, it might well be a price worth paying.

The bottom line is that any debate on whether the spam problem should be addressed via legislation leads to two key questions. First, would the legislation be effective in solving the problem? Second, is the price of the direct and indirect consequences of this legislation worth paying? Obviously and unfortunately, the answers to these questions are unknowable at the present time.

Virtually everyone does agree that it is by no means certain that good legislation will result from any governmental legislative efforts. However, because it is the nature (and, indeed, the vocation) of politicians and lawyers to legislate on topical issues, some laws are almost sure to be forthcoming. And if poor legislation does get passed, it would

---

If an antispam law would have an analogous impact to the junk fax legislation, it would be hard for anyone who opposes spam, no matter how anarchistic he or she might be, not to concede that such legislation is a good thing.
It is absolutely in our best interests (out of self-preservation, if nothing else) both to try to understand the issues and to guide our legislators by written and spoken commentary.

almost certainly be more difficult to undo this and obtain effective legal solutions in the future.

Almost no laws have been passed anywhere in the world to cover spam. It is absolutely in our best interests (out of self-preservation, if nothing else) both to try to understand the issues and to guide our legislators by written and spoken commentary. If the issues involving UCE are important to you, we urge you to educate both yourself and your legislators, regardless of your personal stance.

Here is a listing of the most prominent pieces of pending US legislation and some brief commentary on how we view their relative merits. Some good, if partisan, overview of these bills is available at the CAUCE (the Coalition Against Unsolicited Commercial Email) Web site. The focus of all these bills is on email, not USENET spam.

- **US H.R. 1748** (the Smith bill). This is an attempt to extend the junk fax law to apply to UCE. This bill is the one that most antisam organizations (like CAUCE) support. Of the national legislation that has been proposed so far, this is by far the most virulently antisam, and the one we feel has the greatest merit. Read the text of this bill or read Representative Smith’s commentary on his legislation.

- **US S. 771** (the Murkowski bill). The Murkowski bill would require that each UCE have the word “advertisement” as the first word in the subject line of each message and that ISPs install and filter messages for their customers or face legal actions and fines by the Federal Trade Commission. This means that ISPs would have to pay significant costs to support the spamming infrastructure. Although we don’t particularly want to see this bill pass, at the very least, it would allow easy filtering of spam. On the negative side, it would have further deleterious effects on the performance of email, because of further processing, filtering, and increased volume of UCE due to legal sanctioning. Finally, it places the burden on the consumer to get off mailing lists (the so-called “opt out” provision, which requires a consumer to send mail to the advertiser to be removed from a mailing list), rather than having advertisers ask consumers if they want to receive their mailings. All considered, we cannot support this piece of legislation. Read the text of this bill or read Senator Murkowski’s commentary on his legislation.

- **US S. 875** (the Torricelli bill). This bill has the support of the Direct Marketing Association and other “legitimate” members of the marketing community (technical people always put quotation marks around the word “legitimate” when talking about marketing people) who claim that they don’t condone the sort of spam that has gone on so far, but feel that UCE has enormous marketing potential and should remain open as an advertising option. It explicitly disallows the forging of headers, but includes the “opt out” provision that we are against. It appears to be slightly stronger than the Murkowski bill, but not enough so that it would be truly effective in combatting spam. Although it does have some merit, we oppose this bill. Read the text of this bill or read Senator Torricelli’s commentary on his legislation.

- **US H.R. 2368** (the Tauzin bill). This bill is the weakest of the four. Like the Murkowski bill, it would require identifiable subject lines and have an “opt out” policy. The bill would also create a panel of people to specify responsible Internet marketing practices. The biggest downside of this bill is that adherence to its articles would be completely voluntary. Obviously, this does nothing to help stop spam. We strongly oppose this bill. Read the text of this bill. Currently there is no commentary, but you can examine Representative Tauzin’s Web page.

Of course, even with good legislation, there are problems. First and foremost, these laws would be applicable only within the United States. A major reason there are currently
few problems with junk faxes that originate outside the country is because sending these is prohibitively expensive (although perhaps junk faxes just haven’t figured out how to use the Internet for this yet). Because the Internet currently employs a distance insensitive pricing model, legislative action in this country could simply prompt a migration of the spammers to offshore locations. Not only would this lead to further loading of already congested international links, but it could lead to either an international law enforcement nightmare or, more likely, a situation where nobody can take effective legal action.

In any case, unless the organizations or individuals sending the UCE are held accountable, there’s not much anyone can do about overseas or domestic spamming. For all of these reasons and many more, technical solutions to UCE certainly are easier to implement! And although good legislation may help reduce spam on the Internet, it is our opinion that even very good new laws will not completely solve the problem, and strong technical mechanisms will still be the first line of defense.

Conclusion
For better or for worse, spam is here to stay. Electronic mail and news are each simply too effective a communication tool to be ignored by people wanting either to make money or to spread a message to the masses. However, we are not advocating total elimination of UCE; we simply want people to use responsible and acceptable distribution practices. Abusing resources is not acceptable, and that is what this article is trying to help prevent.

Unfortunately, even if everyone implemented all the solutions we discussed in this article, spam would continue. It is always going to be possible to misuse the Internet because its two main strengths, power and flexibility, are particularly easy to exploit. But we feel that some measures and practices are better than others, and if everyone (or even only ISPs) adopted them, they would definitely reduce the total amount of spam on the Internet. The most significant positive changes are:

- elimination of open mail relays
- strict USENET news backoff algorithms that prevent posters from flooding the Internet
- a significant but realistic fine

The problems caused by spam and UCE are real and significant. The entire Internet has been affected by this malady, but there are things we can do to alleviate the problem. We believe that a combination of existing technical solutions, many of them described in this article, future technical work, and cooperation among Internet service providers can significantly impact network abusers without unduly affecting responsible Internet users. Much work still needs to be done on this, but we have shared our experiences and the techniques that EarthLink Network has found useful in combating these problems, and we hope to start a dialog on how we can further reduce the problems that spamming causes for the Internet as a whole.

Final Note
This article is a work in progress. It represents the version as of early February 1998. Updated versions can be found at <http://www.trouble.org/security/spam_war.html>.

Acknowledgments
We'd like to thank David Beckemeyer for information on his IP Caller ID proposal, Tim Bosserman for providing information on RADIUS Accounting, Dave Hayes for his work on the news backoff patches, Harris Schwartz for information on the results of
our efforts, and Lisa Hoyt for her unceasing work in making the spammers leave
EarthLink Network for some other provider, for now . . .

Appendix
We utilized a new SATAN module that used a very simple method to determine if a
host allowed unrestricted mail relaying. Our methodology:

■ An untrusted/reasonably random system was used to do the testing. Trouble.org has
no special privileges or status with any of the ISPs tested.
■ Nslookup was used to examine the DNS records of the target ISP, and all MX hosts
were collected.
■ The MX host’s SMTP daemon was connected, and the following commands were
issued:

```
heho tsunami.trouble.org
mail from: <zen@tsunami.trouble.org>
rcpt to: <zen@tsunami.trouble.org@target.host>
```

Note that the percent token (%) was used instead of the at sign (@) to determine if the
system was a mail relay. Simple antimal relay rules in the SMTP daemon (and those
proposed initially by sendmail.org) would allow this sort of mail to be delivered; we
found several sites that blocked the latter method but not the former.

The return codes were then examined. If an appropriate response was received (250,
etc.), the host was assumed to be an unrestricted mail relay. Obviously, it would be best
to actually send the mail and see if it was delivered, rather than this partial test, but the
difficulties of scanning arbitrarily large networks from arbitrary hosts make this a more
palatable (at least, significantly easier to program) solution.

The method used is error prone in many ways, however. Although none of them are
fatal, false positives could occur in numerous ways:

■ Incorrect DNS information. Some hosts had no MX hosts listed at all.
■ MX records might point to systems outside the ISPs control.
■ Duplicate MX records. Smaller ISPs are sometimes merely repackaging of a larger
ISP.
■ The SMTP daemons could return a false positive or be non-RFC compliant.

False negatives are also possible, due to the following:

■ Networks or individual hosts could have been down. We didn’t ping the target sys-
tems to see if they were up because of the proliferation of packet screens that block
ICMP requests.
■ The ISP might not even exist. The list we got the targets from, like anything on the
net, is virtually guaranteed to be out of date
■ We examined only the MX mail hosts. If we had examined the entire ISP’s network,
we would presumably have found more mail relays than we found in this survey.

Ideally, we would hope that the errors would either not come up or simply cancel
themselves out, but in practice, results found in this survey are likely an upper limit,
probably within 10% of the final total (this is not an exact science!).
Practically everyone has run across spam, especially anyone who runs a network, posts to USENET, lists an address on a Web page, or just plain has been around and active long enough to have a well-known address. There are almost as many misconceptions, mistakes, and just plain lies running around the scene as there are actual pieces of spam. Herewith, a brief tour of the lies, the legalities, and the pending legislation.

The Top Five Big Lies About Spam

1. It's protected by the First Amendment.

In Cyber Promotions Inc. v. America Online Inc., 948 F. Supp. 436 (E.D. Pa. 1996), the court ruled "Cyber has no right under the First Amendment to the United States Constitution to send unsolicited e-mail." This is the only case to make a serious First Amendment argument in favor of spamming, and it was struck down unequivocally by the court.

2. It's perfectly legal.

The junk fax law, 47 USC 227, most likely does not apply to spam. However, whenever spammers have been sued by ISPs (such as AOL, Compuserve, Concentric, and so on), the spammers have either settled or been enjoined to stop. The problem today is not that it's legal — every court it's been brought before has said it isn't — it's that it takes a civil suit to stop it.

3. It doesn't cost you anything.

ISPs report that from 5% to 30% of all email they receive is spam. The extra capacity to handle that unwanted traffic is paid for out of ISP subscriber fees. When a mail server is crashed because of the spam load, everybody pays.

Further, ISPs are having to dedicate technicians to answering spam complaints. Those technicians are busy handling spam instead of maintaining and improving services, and they are paid for out of subscriber fees as well.

4. It's illegal for your ISP to block it for you.

Mail servers are usually private property. Their owners, be they ISPs, other companies, or government institutions, can do whatever they want with them. The limit is in their contract with their customers, not in what the spammers want. If an ISP blocks spam and customers object, those customers can take their business elsewhere. In most cases, the customers object only if the ISP blocks spam without telling them.

5. Small businesses can't compete on the Internet without it.

First, the Internet does not exist to provide a subsidy to nonviable businesses. If your company can't cut the mustard without stealing other people's resources, it should be shut down.

Second, there are some ten million small businesses in the US today and millions more worldwide. If those businesses all started using email to advertise, everyone's email box would look like your favorite big city classified ads section, with hundreds if not thousands of ads. Spamming simply does not scale up well.
Legal Tactics Against Spam

Criminal Law

No criminal laws are applicable to spamming. Even though spam sometimes creates a “denial of service” condition – wherein servers are rendered totally unusable for extended periods – none of the incidents to date has crossed the threshold that would make them interesting to law enforcement officials. This is a shame because I firmly believe that making spamming a capital crime would cut the problem drastically.

Lawsuits

Several lawsuits have been settled or won in the spam arena. They’ve used a number of grounds.

Conversion. “Conversion” is the civil law equivalent of plain theft. The defendant “converts” some of the plaintiff’s property for his or her own use. Compuserve’s case based its arguments upon conversion. Cyberpromo settled, so the theory has not been fully tested in court, but there is good reason to believe it would hold up.

Defamation/Forgery. Spam usually comes from a forged origin address. If the origin address is real, the owner’s reputation may be adversely affected by the spam. In Parker v. C.N. Enterprises (Tex. Travis County Dist. Ct. Sept. 17, 1997), the court recognized the damage to the Parkers when their domain flowers.com was used in a spam, and awarded damages to the plaintiff.

Actual Damages. In some cases, a computer system may be crashed or valuable, legitimate email may be lost due to spam. In those cases, the plaintiff may be awarded compensatory damages for the loss, as well as court costs. This has happened in a number of cases, including Compuserve v. Cyber Promotions and Parker v. C.N Enterprises.

New Law

Because existing law does not adequately cover spam, a few bills have been proposed in Congress. All address commercial email. Nearly everyone involved in the campaign against spam believes that the First Amendment protects noncommercial spam in the US.

The Murkowski Bill. The Unsolicited Commercial Electronic Mail Choice Act of 1997 (S. 771) would require tags on any commercial email, with ISPs required to offer their users filtering on those tags.

The Torricelli Bill. The Electronic Mailbox Protection Act of 1997 (S. 875) is fairly vague. It would require advertisers to follow some (unspecific) Internet standard, honor opt out (“don’t mail me”) requests, and use valid reply addresses. There’s some fear that it would give IETF decisions in this area the force of law.

The Smith (CAUCE) Bill. The Netizens Protection Act of 1997 (H.R. 1748) simply amends the junk fax law to include email. The protections would include a requirement for valid address information in ads, a private right to collect $500 from an email advertiser, and treble damages for trying to evade the law.

Why Make a Law?

Technical Means Are Failing

The key technical means of blocking spam is filtering – by sender address, IP address range, message headers, or body. Spammers fake addresses regularly, use different providers, and change their messages around precisely to evade filters. Broader filters can catch more spam, but they begin to catch nonspam as well.
Social Methods Require Creation of a Backbone “Cartel”

If all the backbone providers got together and enforced strong antispam Acceptable Use Policies, they could put a huge dent in the amount of spam because they would shut down the remaining spam houses. There are two problems with this concept:

1. The backbones would all have to agree on something, which is unlikely.
2. This might give people the idea that there are places from which the content of the Internet can be controlled.

The Whole World Looks to the US for Guidance

Spam started out as an American problem. The big spam houses are all US-based. Our neighbors on the Internet want us to stop dumping our trash all over them, and the US needs to set an example for the rest of the world on how to deal with Internet governance.

A Good Law Will Redress the Cost Imbalance

The fundamental problem with spam is that it imposes on recipients costs that they cannot recover. A good law, like the Smith bill, will allow recipients to offset their costs of receiving spam with cash awards. A law that simply requires tagging of spam or opt out lists will create an explosion of advertising email with no way for recipients to recover their costs.

The Netizens Protection Act of 1997 is supported by CAUCE, the Coalition Against Unsolicited Commercial Email. I am the chairman of CAUCE, so I have a small axe to grind. Here are some reasons to support it:

- It is based on the TCPA (junk fax law). 47 USC 227 (the Telephone Consumer Protection Act) regulates telemarketing and advertising via fax. Faxed advertisements are prohibited outright unless the sender and the recipient have a preexisting business relationship. The TCPA has been upheld in federal court.

- It does not mandate an enforcement agency. No government agency will monitor email looking for violations. Action occurs only when recipients decide they have received unsolicited email advertisements.

- There is private right of action ($500 per occurrence). The TCPA may be enforced by the states or through a private right of action. An individual may bring action in court for $500 or actual damages, whichever is greater and/or to get an injunction against continuation of the faxing. The Smith bill would bring the same protection to email.

- It goes after the advertiser, not the agency, and US companies can’t use offshore agencies to evade this law. The law targets the sender of the ad. Although some agent may distribute the ad, the advertiser will have to have its contact information in the ad in order to get any business. If advertisers are in the US, the law applies to them.

The Current Situation

The spam war is at somewhat of a stalemate in the legislative arena. The Smith bill is stuck in committee. Meanwhile, at least three states – California, Kentucky, and Washington – have introduced state-level bills to outlaw spam. The major providers and backbones are slowly becoming more aggressive about stopping spam from their networks, but there are thousands of ISPs for spammers to take advantage of and tens or hundreds of thousands of open mail relays for them to abuse.

For information about organized efforts to fight the war against spam, visit CAUCE at <http://www.cauce.org/> or “Fight Spam on the Internet” at <http://spam.abuse.net/>.
Rob Kolstad interviewed Dr. Clair W. Goldsmith via email over the last three
months. Goldsmith has held many posi-
tions in the academic and medical com-
puting communities in the last couple of
decades. Clair was also president of
DECUS for several years.

interview with
Dr. Clair W. Goldsmith

Rob You are a high-level guy at the University of Texas, Austin, computing center,
right? Please tell us a bit about your position.

Clair I have a typically academic title: Deputy Director, Academic Computing and
Instructional Technology Services at the University of Texas, Austin. UT Austin has sep-
arate computing centers for academic and administrative computing. The differences
are more apparent than real. For example, we are working to combine our Help Desk
functions – academic computing has a much larger effort for this than administrative
computing. We also use the same communications network infrastructure. I often deal
with the infractions that occur on the administrative computer system as well.

Rob And UT serves a large user community? Lots of hardware?

Clair There are lots of measures of that. There are 75,000+ users who publish their
email address in our X.500 directory. Probably the most interesting statistic is that there
are over 34,000 computers on the campus network. Or we get about 13,000,000 hits per
month to our Web site, which is 250,000+ pages on 300 servers.

Rob What sorts of interesting issues do you confront?

Clair All of the complaints about the use of technology can be sent to me via
<abuse@utexas.edu>. Although issues involve spam, commercial use, forged email, and
harassment, the most difficult and interesting cases have to do with freedom of speech,
academic freedom, copyright, and occasionally privacy.

Rob So you have to deal with all the random acts that students might commit
when they can use the net to communicate via email, Web, or other standard tech-
niques?

Clair Yes, and also faculty staff and their dependents (if the resources are being
shared – which is always inappropriate and illegal under Texas law). In one case, a par-
ticularly colorful stream of profanity was sent to someone outside the university, who
complained. When the staff member got a letter with the complaint attached, he called
me to apologize. He seemed very surprised that his 14-year-old son knew those words.
I also had a case where the son of a staff member advertised video gaming equipment
for sale, cashed the check, and refused to deliver the equipment, saying that he was only
16 and could not be held responsible. As it turns out, his parents can. The father got the
email about this on the first day of a month-long family vacation in Hawaii.

Most recently, we have had problems with students setting up sites that distribute
the intellectual property of others: games, recorded music, animation, and software. Most
cases are handled by putting the student on disciplinary probation. However, in two
cases we have suspended students: one for selling access to pirated software and one for
repeated distribution of recorded music.

I also have had the police get a search warrant for a dorm room because a student had
constructed a trojan horse to obtain the login identifiers and passwords that he pub-
lished in an alt. newsgroup.

Rob What happens when a student, faculty member, or staff member puts up a
"hate" page (anti-Semitic, homophobic, etc.)?

Clair I usually get a phone call or email complaint, to which I reply that we will
review the information supplied and determine if any rules or laws have been broken.
However, the examples you gave do not break any rule or law. In fact, because we are
state assisted, we must not abridge the First Amendment right of free speech and there-
fore cannot prohibit such. Private institutions might have different rules – but as a state institution we must follow the laws carefully.

Rob What about dirty pictures? What about really dirty pictures?

Clair Then the Web page has really dirty pictures on it. Seriously, the First Amendment requirement is very strong. And there is no definition of “really dirty pictures.” As long as it’s pornography, it is protected speech. There is a definition, of sorts, of obscene: offends local sensibilities. Thus, although I have seen some things that make me nauseous, there has never been anything that we have taken down because of content.

Rob What happens when a student sends out 100,000 spam emails promoting some random product or thought?

Clair Well, if it’s a product, they get a mail message reminding them they cannot use state property for commercial purposes and there is a prohibition against spamming. If it is only a thought, then they just get the spamming reminder.

Rob That’s it? No special “appropriate usage guidelines” or anything? Doesn’t that cost you a lot of resources?

Clair We have what we call “responsible use guidelines.” They can be found in “Looking for Trouble?” at <http://www.utexas.edu/cc/policies/trouble.html>. However, we do define a portion of the resources for each use – that would simply be unmanageable. However, we do have a rule against spam, and when it is broken, we do contact the individual. Most are surprised that we noticed and then are subsequently embarrassed. We do not often have repeat offenders; and when we do, it is likely to indicate a more serious problem for which stronger remedies are available. In any case, I do not have to worry about discipline for students, faculty, or staff. I only produce the evidence chain.

We believe that attempts to control use will be ineffective in this environment. It is better to set standards of behavior and use and measure the complaints against those standards.

Rob How do you go about searching students’ rooms to get evidence to help the authorities?

Clair Only on invitation. Normally, we do all investigating via the Net or through other records and logs. Actually, we would never go into a student’s room. The police do that – and only with permission or a warrant.

Rob So, being 18, the students have the complete set of rights we might all expect as citizens, and you have to make the call each and every day as to whether they have crossed the thin line that divides “protected” activities from illegal ones?

Clair Yes, the serious issue for students is that they be protected appropriately. They sometimes think that because they are students, rules and laws don’t really apply. They are dumbfounded to learn that they do. A case in point is the student who created the trojan horse. He didn’t think anyone would notice his posting of accounts and passwords. This is against the rules and state law. When the police woke him and his roommate at 7:00 am, the first thing he said was, “I bet you’re here about the accounts I posted.” He learned this statement is admissible, even though he had not yet been Miranda-ized, because he was not then a suspect.

Often, finding the evidence is an interesting problem. I recently had a complaint by a computer science undergraduate student that his account in CS had been hacked. CS had blocked his account and although they found him believable, told him it was his

Dr. Clair W. Goldsmith

“We believe that attempts to control use will be ineffective in this environment. It is better to set standards of behavior and use and measure the complaints against those standards.”
problem to find the culprit. By the time he got to me, he had been to the source of one of the attacks, microbiology, and had the account number of the person who was logged on at the time of the attack. Under federal law, if the account owner is a student, I cannot release the name of that student. Therefore, I investigate the incident, identify the facts, and turn the information over to Student Judicial Services. In this case, a second break-in had occurred from a chemistry research area, where no account is necessary to use the computers. The attack occurred on a holiday, so it was clear whoever did it had a building key. The faculty member responsible for the area provided a list of students and staff with access. There was no match to the person who had attacked from the microbiology computer. After thinking about it for a day, I queried the X.500 directory on the telephone number of the microbiology suspect. A name on the Chemistry list did show up—his wife, it turns out.

**Rob**  Do you get a lot of flak from people outside the university who do not understand the rules?

**Clair**  Yes. Actually, once explained, there is not much push-back. Typically, it will be from other institutions who know better and try to see if they can get us to give up information. We do not give out information on students to police authorities without a subpoena or, in the case of clear and obvious need, such as a disaster or other life or death matter.

I once had a person at a premier Ivy League institution, who should have known better, tell me that there could be no such law that required the permission of students to release information about the student to him. After several email exchanges, where he repeatedly told me I was wrong, I sent him the appendix from our General Information catalog that contains the text of the federal law.

**Rob**  How much time could it take to deal with all these sorts of problems?

**Clair**  This is almost a full-time job. I also have a quarter-time law student who deals with routine cases. A trivial case, such as one the law student deals with, takes about 30 minutes to respond to the complainant and ask for the account number used. This may require additional effort, such as requesting the complete email headers from the complainant or searching a newsgroup archive for the complete headers. It takes an additional 45 minutes to complete the information necessary for either a referral or informational referral to Student Judicial Services.

**Rob**  Do you have to deal with incoming spam and hate mail? What happens?

**Clair**  I wrote about this in our November newsletter, see [http://www.utexas.edu/cc/newsletter/nov97/spam.html](http://www.utexas.edu/cc/newsletter/nov97/spam.html). We do try to minimize spam by not accepting email whose domain names are not resolvable—upwards of 40,000 daily—and we exchange lists with others about known spam producers.

As for the hate email, this does come under the heading of protected speech. There may be cases where it is harassment, but that is a judgment made by the recipient and Student Judicial Services.

For Web pages maintained by terrorist groups, the only effective deterrent I have seen was creditable death threats to family members of the Web page owner. Unfortunately, this may be considered "terroristic threats" that are against state law, but it was not reported as such.
Rob What’s the most rewarding part of your job?

Clair There are a couple of aspects that are truly rewarding.

First, when someone has received (death) threats and is genuinely upset, I can explain what is likely going on, what the person’s options are, and that they are not totally in the control of someone else.

Second is dealing with those outside the university to explain what our rules are, how they are enforced, and why we have the rules. Most people really do appreciate the explanation and come to understand our position, even if they do not completely agree with it.

I very much enjoy this role of being a point of human contact for those who believe themselves injured. And, strange as it may sound, balancing that with ensuring that the accused has all the evidence so that whatever discipline incurred is justified.

“If you are interested in the technology policy issues in education, government, and industry,” Clair says, “the most comprehensive Web source, I have found is Computer Policy & Law Web at Cornell University” <http://www.cornell.edu/CPL/Policies/>.
Asides:
Linux had a different heritage; it started as Minix from the mid-1980s. See <http://www.cs.vu.nl/~ast/minix.html> or <comp.os.minix> for details.
William Jolitz did much of the early porting work to the Intel 386 architecture. See "Books and Articles" on page 58.
I am glossing over many historical details. Look at Peter Solus, A Quarter Century of Unix (Addison-Wesley, 1994) for in-depth history.

they could get access to parts of SunOS.) No longer could we dig in to see what was happening, fix things, or make local customizations. One consolation to computer users was that throughout the 1980s, computers steadily increased in performance with the introduction of new Sun, DEC, SGI, and other workstations. But on these systems, source code was still not generally available.

By the early 1990s, the PCs started offering performance comparable to workstations. I, for one, believed that the x86 architecture would be eclipsed by the RISC CPUs, but Intel proved me wrong. The cost advantage of a PC balanced out its ugly packaging and clunky integration. But PCs were stuck running primitive DOS software. Who would want to run a system that crashed frequently and didn’t offer memory protection, multitasking, or virtual memory? It was looking bleak for progress.

A key enabling event happened in the early 1990s. The folks at Berkeley prepared and released an unencumbered source code UNIX to the public. AT&T (and its UNIX successors, USL and Novell) claimed that the code could not be publicly released because it contained material originating from AT&T. It turned into a multi-year legal battle, finally resulting in the decision that the fruits of 20 years of cooperative Internet (ARPANET) work, now known as 4.4BSD, could be released to the public. The importance of this event to the source code UNIX community is huge. None of BSDI, FreeBSD, NetBSD, or OpenBSD would have been possible if Berkeley and friends hadn’t made the effort to get their work released.

We’ve come full circle with source code UNIX; now – like 20 years ago – you can have the full-featured operating system, complete with source code for all of it. But today you don’t have to timeshare the system; you can have source code UNIX on your own inexpensive computer.

Over a number of months, this column will explore the benefits of running source code UNIX on a commodity PC computer. When concrete examples are necessary, I will often draw from my experience with FreeBSD, but most of my content will apply (often exactly) to other versions such as BSDI, OpenBSD, Linux and NetBSD. I will not debate which version is “best,” because all versions excel in most areas and it is not worth arguing over trivial details. Upcoming columns will deal with choosing hardware, performance of the I/O system, building a Web server, building a firewall, publicly available[1] software and customizing source code UNIX for embedded products. I am most willing to listen to feedback and suggestions, but please, no comments on which version is best.

Why Would You Want To Set Up A Source Code UNIX PC?
I imagine a number of the readers are quite content with their traditional UNIX workstations. You probably have a Sun SPARC running Solaris, an HP with HPUX, an alpha with Digital UNIX, an SGI with IRIX, or an IBM with AIX, etc. All of these are completely functional systems; however, they all lack source code. Do you have custom hardware for which you need an operating system? Are you curious how a hardware device works? Do you want to look at the virtual memory system? Do you frequently wish you could slightly change the way some of your vendor-supplied applications work? Do you need to fix something that your vendor won’t get to for six months? Do you need to write an application that could be leveraged from an existing application? Does the documentation hint of a possible method to solve your problem, but the details are missing? If you could only glance at the code, the solution might be clear. Well, you’re generally out of luck with the above vendor combinations.

Source code UNIX offered me the most expeditious method to deliver a sophisticated, feature-embellished product based on commodity PC hardware. At my former employ-
er, I was in charge of designing and implementing the software for a realtime product that among other things, played and recorded digital video at 30MB/s to our RAID-3 subsystem. The box consisted of a Pentium single-board computer, a custom CCIR601 video board, and a custom mother board that contained 20 SCSI busses, some fast memory with an XOR engine, and an Ethernet chip. We also had a bunch of miscellaneous devices such as temperature sensors, fan monitors, and LED indicators for which we wrote “driverlets.” We considered some of the commercial realtime operating systems, but their source code wasn’t available. Our naive hardware engineers pushed for an NT-based solution, but they didn’t understand what it takes to get a system running without source code. If we had gone with NT, our little company would still be begging Microsoft for modifications, patches, and cooperation. Instead, we controlled our own destiny, and within days of the hardware functioning, UNIX controlled the product.

Using UNIX made so much sense because we required most of its functionality. The product needed networking (TCP/IP), a filesystem, field update capability (CVSUP), Graphical User Interface (Tcl/Tk), realtime process management, virtual memory, and more. With six software engineers, it would have taken much longer than 12 months to re-implement all of this work.

Source code UNIX systems are for engineers who need (or want) to have more control over their system than the vendors allow. It is also the best way to learn about how operating systems in general and UNIX specifically work.

**Source code UNIX systems are for engineers who need (or want) to have more control over their system than the vendors allow. It is also the best way to learn about how operating systems in general and UNIX specifically work.**

Source code UNIX systems have various methods for configuring third-party applications. For instance, there is the FreeBSD ports tree and the Redhat Package Manager (rpm) for Linux. These facilities make it simple to install the latest versions of thousand of applications that come with source code. Some examples are audio tools, mail systems, Web utilities, databases, and graphics packages. I’ll dedicate a column in the future to reviewing ports and packages.

**How About a Windows Server?**

Ironically, source code UNIX makes an excellent disk, mail, and print server for Microsoft Windows 95 (Win95) clients. After you install Samba,[2] Win95 PCs can easily access mail, Web pages and files that reside on an inexpensive, old, low-end PC. In this environment, you can set up a powerful backup capability for the clients. You have an alternative to the NT server; something that is much less expensive and performs better in many situations. Users who have Win95 will be happy, and you, the administrator, will be happy, too. (The Netatalk package gives these capabilities to Apple Macintosh clients.)

**Are There Economic Reasons?**

There is an economic reason to consider source code UNIX. Even though the traditional vendors have rapidly dropped their workstation’s prices, commodity PC hardware is still “dirt cheap.” The technology has been and will most likely continue to be one step behind with PCs, but it’s really hard to beat the price/performance ratio. If you are willing to be careful when shopping and avoid certain problematic hardware, you can put together a solid, high-performance system for a small amount of money ($1,000-$2,000). Of course, it is reasonable that traditional UNIX vendors get a premium for their carefully integrated and balanced systems. They also get the premium for support
that most importantly includes technical support—so you don’t need a guru around. If
you can’t support your system or you don’t want to do any of the integration work,
then source code UNIX may not be for you.

Instead of buying a new PC, a scenario for running source code UNIX is to find “obso-
lete” PC hardware—the stuff that no longer can efficiently run Win95 or NT. Many
consider a 16MB 486 underpowered to run Office97. Certainly, NT will require at least
a husky Pentium with 32MB or more. But source code UNIX runs on these “discards.”
If you don’t need an X11 system, an 8MB 486 system can be peppy for certain appli-
cations. You don’t need much disk space for source code UNIX. A small, couple hundred
MB disk is big enough for a substantial development system.

Keep in mind too that the cost of running Windows on your PC is not trivial. After you
buy the operating system for a couple of hundred dollars, you still have to buy every-
thing else. Plan on spending up to $500 for a C development system. Spreadsheets and
word processors cost a few hundred dollars. Then you might need a mail system, virus
checker, Web server, security software, and a backup system—every one costing a lot of
money. By the time you are done, you could easily spend $1,000-$2,000. All of these
utilities are included in the source code UNIX systems. The free versions might not be
as feature laden, but you can certainly accomplish the same tasks efficiently. You also
have the option of purchasing commercial software such as spreadsheets or word
processors for your UNIX system.

Finally, you may have an existing system that is already running MS Windows. It is
straightforward to split the disk into a Windows system and a UNIX system. It takes a
minute or two to switch between the operating systems with a reboot. I run one of my
systems with 500MB for Windows and 1.5GB for source code UNIX. My source code
UNIX system mounts the Windows partition, facilitating file transfer between the oper-
ating systems.

What Would a Modest New PC Cost?
If you haven’t looked recently, you will be amazed how fast PC prices have dropped. A
very respectable new system without monitor costs under $700 (February 1998). It
consists of the following:

- 166MHz Pentium
- 16MB memory
- 2GB disk
- 16x CD-ROM
- 56K modem

Monitors range in price from $200 for a cheap 14-inch unit to over $1,600 for a high
quality 21-inch monitor. A nice 17-inch color display costs between $500 and $600.

What If I Want the High-End Stuff?
A hot 300MHz Pentium II MMX with 64MB of memory costs about $1,400 (February
1998). For that price, you would also get an 8GB disk, a DVD CD-ROM, a 56K modem,
and a 4MB video board, but no monitor.

What If I’m on a Tight Budget?
With just a little hunting, you should be able to find a complete used system for $100-
$500. A used Pentium 90MHz with 16MB memory, 1GB disk, and monitor should be
easy to find. The 486 systems should be almost free, but with monitor, disk, and printer, maybe $100 or $200. Many companies mothball these systems because they cannot run the current Windows software. You can exploit this fact to gather cheap or free hardware that makes good firewalls or file servers. But for development, I’d want a bit more pep than a 486.

**What Source Code UNIX Should I Run?**

(I’m going to try hard to avoid religious wars!) First, a disclaimer: I think all of the source code UNIX systems are viable – no need to fuss. My history is with FreeBSD, which will show when I give concrete examples. I wish the various source code UNIX groups worked more closely together. I think Jordan Hubbard, a core member of FreeBSD, sums it up well when talking about Linux and FreeBSD:

> So now, within a very short space of time, we’re almost spoiled for choice in having machines several times more powerful than the first multiuser VAX machines and available for under $2,000, and we’ve got not one but several perfectly reasonable free operating systems to chose from. We are in a comparative paradise, and what are some of us doing? Complaining about it! I suppose too much is never enough, eh?

As to which is “best,” I have only one standard reply: try them both, see for yourself, think for yourself. Both groups have given you something for free, at considerable personal effort, and the least you can do is give them the benefit of exerting enough effort to try out what they’re offering before passing judgment (or worse, blindly accepting someone else’s).

Whichever you run, you’re getting a great deal – enjoy!

The source code UNIX systems have a tremendous amount of overlap. They share lots of their innovations. Most of the publicly available applications run fine on every system because the system call interfaces are compatible. The differences are relatively minor and difficult to sum up in a few words. Linux is the most widespread, with probably several million installations worldwide. Linux is often described as Sys V-like and has the reputation of supporting some of the more obscure PC hardware. BSDI offers its commercially supported version that is available with source code. FreeBSD has spent a lot of time tuning its I/O and VM subsystems. They claim to have the busiest public FTP server in the world: [ftp.wcarchive.com](http://ftp.wcarchive.com). NetBSD’s mission is to run on many platforms: PC’s, SPARC’s, alphas, etc. See [http://www.netbsd.org/Ports/index.html](http://www.netbsd.org/Ports/index.html) for the full list. OpenBSD is a recent split from NetBSD. They run on multiple platforms and are committed to fixing security bugs. See [http://www.openbsd.org/goals.html](http://www.openbsd.org/goals.html).

**How Do I Get Started?**

There is a wealth of information available for source code UNIX. I suggest you become familiar with the Web pages for the version you select (sites given in the sidebar on page 56). These home pages are the definitive source for the various systems and may supersede information provided in this article.

Subscribe to some of the USENET groups listed below. Get the Frequently Asked Questions document:

```
ftp rtfm.mit.edu
cd pub/usenet/news.answers/386bsd-faq
mget part*
```

There are a number of ways to load your system with source code UNIX. You might be able to clone someone else’s disk. If you have good Internet connectivity, you might
download the system over the wire. (And once you have it installed, you may want to incrementally update the code base with programs such as CVSUP, HRC, SUP or Anon-CVS. More on this in future columns.) But for most of you, I recommend spending $30-$50 for a CD distribution set. (Note, you are paying for the convenience of having your own CD disks. Feel free to borrow someone’s CDs or lend yours.) From these disks, installation is usually quick and easy. Next month I’ll go through the typical steps for loading a system. I’ll also spend some time discussing various hardware options. For now, spend some time on the Web, get familiar with some of your options, then choose a starting source code UNIX system and order its CD-ROM set.

I’d like to thank the following reviewers: Ken Merry, Steve Gaede, Mike Durian, and Joel Rem.

**Books and Articles**

For those who want to understand the organization of BSD kernels, you will want Kirk McKusick et al., *The Design and Implementation of the 4.4BSD Operating Systems* (Addison Wesley, 1996.)


**Notes**

[1] I define publicly available software as software with freely redistributable source code.

I may have found the notebook of my dreams. It weighs less than two pounds, has a gigabyte of hard drive and an 800x600 color display – and it runs UNIX. I saw this notebook, a Sony PCG 505, during the USENIX Security Symposium in San Antonio. Although this notebook does not have the battery life I would like, the other features (including one PCI slot for a network interface card) are just fine. At the time I write this, it is available only in Japan and costs ¥250,000. At the current exchange rate, this is about $1,600US.

Nope, this is not an April fool. But it’s still not exactly what I wanted either. Battery life is too short, screen is the right size for a single xterm window, which would be easily readable. And the keyboard is reduced in size to fit the smaller form factor. I have decided to set my sights on something rather different that exists only as prototypes and experimental models yet. The wearable computer.

### Wearable Computers

Wearable computers have been around for a while, with the best-known hotbed of users at MIT (<http://lcs-www.media.mit.edu/projects/wearables/>). Wearers of these computers have been nicknamed the Borg, partly because the currently popular display unit, Private Eye, covers up one eye (reminiscent of the very plugged-in Borg of Next Generation). A true wearable computer is always on, so requires a very hefty battery as well. Batteries are worn in a fanny pack, along with several PC boards, power inverters, a few hardware ports for jacking in, and hard drive, and weighs (I am guessing) about ten pounds with a battery life of eight hours.

Some of the more innovative designs include GPS and cell-phone modems, so the wearers always know where they are, are always connected, and other people on the network can find them (in both real and cyber spaces). Steve Mann has added a video input device, so other people can see what he sees. He can scan a person’s face and have his software supply you with his or her name. You could have the GPS not only locate you, but provide directions to your destination. (This feature might prove a big seller within the Pentagon and other mazelike buildings.)

I must confess that I am not ready yet to be wired 16-18 hours a day. I do not even carry a pager or a cell phone (yet), such is my yearning for the illusion of freedom. Yet a lot of the technology of the Borg could find itself in more conservative designs. For example, the preferred “keyboard” for a wearable is the Twiddler, a one-handed chording keyboard that includes the mouse. It was actually the Twiddler that got me started on this thread. The Twiddler can be used in the left or right hand and has three columns of four buttons for the fingers and a circle of six buttons for the thumb. With a thumb button depressed, hand movements generate mouse movement. Meanwhile, your hand never leaves the “keyboard.”

But what about chording? I don’t have a Twiddler yet, but did get my hand on an older chording keyboard, known as a BAT. The BAT connects to the keyboard port of a PC compatible and has only seven buttons – four for fingers and three for your thumb. I started through the tutorial and found that I could quickly learn the basic alphabet and start typing. But the BAT is big, and the chording sequences are clumsy. The single finger chords are for the letters “wiry,” and does not include the most commonly used letters (such as “eatr”).

When I mention chording, most people respond by saying, “I already know how to
Although it was the Department of Defense that started the interest in hands-free input and head-mounted displays, I think there will be a booming consumer market for this is less than five years.

type. Why learn something new?” The why is that it leaves a hand free, it is more efficient (ever watched a court reporter chording?), and it may prevent carpal tunnel (no weird wrist position). When Doug Englebart demonstrated the mouse to Steve Jobs, he was using a chording keyboard with the other hand.

Okay, let’s imagine that the Twiddler, or something like it, has replaced both the keyboard and the mouse. We have eliminated about half the requirement for real estate on a notebook and are left with the battery, motherboard, ports, and display. So let’s get rid of the display.

The Private Eye uses a vibrating mirror to present the illusion of a 15-inch monochrome monitor with a resolutions of 720x280. The display unit blocks one eye. The next generation Private Eye, the P7, will have 640x480 resolution with 12-bit color. What I found much more interesting is a newer technology that bounces an image off the lens of a pair of glasses. Only the wire trailing from the glass frames, and a rectangular light spot on one lens, betrays the display to others or blocks your view of the world outside. You get to see three dimensionally. And the resolution is better than the old Private Eye.

So we have now eliminated the keyboard and the display as large, bulky power and space consumers. You could have a box half the size of current notebooks, a display that will never be crushed when the person in front of you reclines the seat, and a keyboard/mouse that keeps one hand free. Lower power requirements translate into longer battery life, and the smaller unit weighs less as well. I think this is getting closer to the notebook computer of my dreams. Perhaps it can include a CD-ROM drive so you can listen to music CDs as well. And forget the floppy – use the network, Luke.

I have written previously that I want to be living in the future now. A discreet, head-mounted display and one-handed keyboard/mouse seems like a big step closer to me. Although it was the Department of Defense that started the interest in hands-free input and head-mounted displays, I think there will be a booming consumer market for this is less than five years.

Acquisitions

Digital Equipment Corporation has agreed to be acquired by Compaq. I was stunned when I heard the news. How far the mighty have fallen. Or perhaps I should say arrogant?

DEC was once the renegade, the developer of “mini” computers, when mini meant small, instead of the mid-size connotation it has today. King of its market niche, DEC became a real power in the late seventies and on into the eighties. But DEC, and its CEO, Ken Olsen, didn’t believe that the coming of lower-priced (and lower-margin) UNIX servers would eat DEC out of its home.

For many years, it had seemed that everyone I had met who worked for DEC had gone elsewhere. Many restructurings had, at long last, made DEC profitable again. DEC still has a stronghold in manufacturing with its VMS operating system running on alpha servers. And DEC has a strong position in the service sector, and something else Compaq has long craved: real presence in the high-end server market.

Together with DEC’s service organization, Compaq might soon make real headway in the large server world, which means, of course, more NT and less UNIX and VMS.

Then customers of DEC or Compaq can do one-stop shopping for everything from the desktop to the mainframe class machine.
Or so the reasoning goes. Just keeping DEC alive has been a monumental task. Compaq CEO Eckhard Pfeiffer looks a little like an aged Clark Kent to me, and he will surely need his Superman alter persona to pull this one off. At the very least, he will cut some product lines from DEC (storage and the money-losing PC and laptop lines). I am glad to be watching this from the outside, and worry about the people I know who still work for DEC.

**Personalities**

I taught NT security for the first time last week. I must admit I really sweated it, because I am not an MSCE (for sure) or even an administrator who runs a domain with 10,000 NT workstations. I have learned that NT does have some interesting and powerful security features. It also has enough complexity to require serious expertise to keep it secure and yet still permit operation by nonadministrators.

I learned something else, too. Although NT has lots of the cool stuff I discovered in UNIX, what it doesn't have is personalities. UNIX had, and has, well-known people who wrote it, added interesting utilities and features, and stayed around to keep them working. Even people who worked for AT&T, such as Thomas, Kernighan, Ritchie, Lesk, Korn, etc. appear as individuals, not invisible programmers working on a profitable project.

Our community recognizes the value of contributed software, such as Perl, Tcl, Apache, Linux, and many other projects, and it is through individual effort, often unpaid, that we have reached the point where we are today. I don't think any of us are willing to give this up. I certainly plan on doing what I personally can to contribute – even if it is no more than writing and teaching.
using java

Write Once, Run . . . Where?

Introduction

Java is intended to be a portable language. Sun makes so much of the phrase “Write Once, Run Anywhere” that they claim it as a trademark. But is Java really that portable? The answer turns out to be “almost.” This article describes my experiences working on the Network Flight Recorder user interface.

What Version?

The first question you have to ask when dealing with Java is “what version?” When we began in June 1997, the prevalent version of Java was 1.0.2. We considered whether we should write in Java 1.0 or Java 1.1 because we knew 1.1 would be available “real soon now.”

Since then (only eight months ago as I write this), two new versions of Java have been released, and both made substantial changes to the standard libraries. In Java 1.1, they changed the GUI event-handling model. Java 1.2 is making still more changes to the user interface components.

Our user interface was intended to be Web-based, which meant we had to have a browser that could execute whatever version of Java we picked. At the time, Netscape 3 and Internet Explorer 3 were both widely used. They executed Java 1.0, so we wrote to Java 1.0.

We discussed writing to Java 1.1 to be used in Netscape 4, which was also due out “real soon now,” but decided it would be difficult to develop for a platform that did not yet exist. Now Netscape 4 and Internet Explorer 4 are available, and they can run Java 1.0 or Java 1.1 code. So how do you choose a Java version?

There are really two choices:

1. Be in a position to dictate which browser your users will run.
2. Use a version of Java that runs in a browser that is widely used by your customer base.

Even though these newer browsers are now available for many systems, not everyone is going to rush out and upgrade immediately. Some people have things to do other than try to keep up with the latest version of everything. Some people have well-considered policies to be cautious about upgrades, based on the theory that upgrading means trading the bugs you know about for all new bugs you have never seen before.

NFR chose to stay with Java 1.0, rather than trying to sell a product while saying, “Yes, it’s Web based, but it won’t work with your browser.”

Following this reasoning, the rest of this article relates to code written for Java 1.0, though it also applies to 1.0 code run in 1.1 environments.

How Portable Is It?

Java is much more portable than C. Assuming that you write “pure Java,” the core features of the language really do live up to the promise of “Write once, run anywhere.” I’ve never seen any instance of basic functionality that is different from platform to platform.
The real portability problems show up when you try to write something graphical. Because Java is being pitched primarily as a language for GUI-like programming, I found this somewhat surprising.

The problem occurs because Java does not implement GUI components directly. Instead, it uses “peer objects.” These peer objects are constructed from the native GUI components for the system you are running on. For Netscape, somebody wrote a peer object that uses Motif buttons. For Internet Explorer, somebody wrote a peer object that uses Microsoft Windows buttons. Initially, this looks like a very elegant solution, but in practice it turned out to be a mistake because it did not work well when it was handed off to third parties to implement.

In your program, you just use java.awt.Button to get a button, so your interface is portable. In principle, you never have to know that there is a Microsoft Windows button that implements that object. In practice, though, the objects do not all behave the same. The incompatibilities fall into two different categories that are sometimes hard to distinguish.

Some of Java’s portability problems come from the poor documentation. The AWT documentation does not always clearly state what an object promises to do.

For example, consider java.awt.Scrollbar. According to the documentation, AWT sends a “scroll absolute event if the user drags the bubble.” But when? It doesn’t say. In fact, the Netscape version of java.awt.Scrollbar sends a scroll event for every mouse move event. The Internet Explorer version of java.awt.Scrollbar does not send a scroll event until after you release the mouse button.

Each author could make a reasonable case that his or her implementation complies with the documentation. After all, the differences exist only in the unspecified area. Only the user who looks at both implementations will get confused.

The other major problem is that some things just don’t work. To demonstrate a few examples of the problems, I put together a small applet that creates a window with some various objects on it. It is available at <http://www.usenix.org/publications/java/usingjava10.html>.

Here are a few examples that you can demonstrate with this applet:

- Some graphical objects do not correctly let you set their colors.
  - Internet Explorer 3/Windows. Several objects do not take their new colors correctly. For example, Buttons will always show black text on a gray background. Scrollbars will show the background color in the slider area, but will always have a gray slider and black arrows on a gray background.
  - Netscape 3/Windows. This is much the same as IE3, but a few objects change colors that did not in IE3.
  - Netscape 4/Windows. As far as I can tell, it misbehaves identically to Internet Explorer. I wonder if they are now calling into DLLs that belong to IE.
  - Netscape 3/UNIX. You can set the colors of most objects other than Choice. Oddly enough, the basic component shows as black on gray, but the popup area shows in the colors you selected.
  - Netscape 4/UNIX. This works about the same as Netscape 3 on UNIX.
  - appletviewer/Solaris. As you might expect, this works pretty well. I have not found any discrepancies.
HotJava 1.0/Solaris. This works about as well as appletviewer, at least as far as colors of objects are concerned.

- When you can set the color of an object, not all environments handle it the same. In most browsers, the effect takes place immediately. In Internet Explorer, the color of the object does not change, but any area of the window that is covered and then exposed will be redrawn in the new color.

- The slider in the scrollbar is supposed to represent the proportion of the displayed area to the total area. In Netscape 3 on Windows, the slider is always the same size, no matter what you try to tell it.

- Expose events fail to work reliably in several environments. The general case works, but there are odd events that should cause redraws but do not. In Netscape and appletviewer on UNIX, many objects lose their expose events if two corners of the object are exposed at the same time. If you cover a Java window with two other windows, as shown in Figure 1, the button will fail to refresh when the java window is raised. It appears that the refresh events never make it to the Java code.

- Some systems can’t resize windows correctly.

Netscape 4/Windows. Sometimes it fails to draw scrollbars correctly when a window is resized. If you cover and expose the window after the resize, the scrollbar comes out right.

Internet Explorer 3/Windows. The Panel.resize() method does not work at all. All your new windows appear at the same size, no matter what you tell them.

I have also encountered some other odd problems. For example, the NFR user interface displays only an empty window when run in HotJava. This was surprising, because I thought that HotJava would be the one place where everything would work. I believe I could have found the problem, but at the time I was also evaluating HotJava and found that it was incredibly slow as a browser. I did not try very hard to find the problem, assuming that not many people were using it.

What To Do?
With the current set of runtime environments, writing portable Java is very much the same as writing portable code in other languages: there are lots of little annoying things that differ from platform to platform. You have to write code that works on all of them.

It is important to remember that it does not matter which environment “works” and which is “broken.” What matters is that there are differences between them, and you have to account for that.

Java has no conditional compilation, so you can’t easily compile shared code with minor differences for different platforms. It doesn’t make sense to the Java mindset – you don’t know the target platform at compile time. You might have your Web server send different class files based on the user’s browser type, but that seems likely to result in its own set of problems.

java.lang.System contains a method to ask about system properties such as the Java version number, a vendor-specific string, and the operating system name. You would think you could ask for the vendor-specific string and implement workarounds based on the environment you are running in, but that doesn’t work. It is a security violation to read the system properties.
You can get much of the same information by having your Web server tell you. As part of HTTP transactions, the browser sends a version string that is available to CGI-bin programs in the environment variable HTTP_USER_AGENT. If you really wanted to, you could fetch that data out and then pass it in to the applet as a parameter.

```
#!/bin/sh
# ta.cgi - starts the ta applet, telling it what browser you have
echo 'content-type: text/html'
echo ''
echo '<applet code=ta codebase=/java height=100 width=100>'
echo '<param name="browser" value="$HTTP_USER_AGENT" />'
echo 'Bummer - you need a Java-enabled browser.'
echo '</applet>'
```

The easiest solution, though, is to stick to a minimal feature set that works in all your target environments. For example, you don't have problems with setting colors if you always use the default colors.

In any other language, you would write a portable program by testing it on all of your desired target platforms. The same rule applies to Java.

This is the solution we used in the NFR user interface. The original design had an interesting color scheme. After finding out all the different ways that objects fail on different platforms, it became apparent that the best solution was to use the default color scheme.

Of course, it would always be possible to write your own Button, your own Scrollbar, etc. This is a viable solution if your time is not valuable, but it doesn't seem appropriate in a commercial development effort.

**Keep in Mind the Alternatives**

We want "Write Once, Run Anywhere" to reduce work for the programmer. Ultimately, the question is less whether "Write Once, Run Anywhere" is a perfect reality as it is whether it is better than the alternatives.

The NFR user interface consists of about 20,000 lines of Java. It could easily take substantially more to write an equivalent user interface in a more conventional language like C.

If I were to write in C, I would have to write the user interface twice – once for UNIX (with all the normal UNIX-UNIX portability issues) and again for Microsoft Windows. Surely I could share some code between the two with careful design, but most of the work would be in the nonportable GUI area.

By comparison, the same Java code implements the same GUI on both platforms. There are some minor portability issues in the GUI area, but these are not nearly as severe as the difference between the X and Microsoft programming models.

**Conclusion**

Apart from the GUI components, Java is highly portable. I have not found any fundamental features that behave oddly on different platforms. The GUI components themselves have problems, but you can still write highly portable programs if you are willing to test on your various target platforms and avoid or work around portability issues. It has not yet reached the ideal, but the amount of work attributed to "porting" is substantially less than common alternatives.
Counters for Your CGI Programs

Usually in this column I demonstrate stand-alone CGI programs that you can drop onto your own UNIX-based (of course) Web site, but this time I’m going to offer you a helpful snippet of code instead — one that addresses a common desire on Web sites: counting visitors.

There are a lot of GIF-based counters available, including my favorite, “wwwcount,” which can do just about everything from wash your sink to polish your car (well, almost). But they don’t help you add your own counter to existing CGI programs; they’re standalone applications that you have to install separately.

You can also use server-side include directives to get page counters if your system is set up correctly. These SSI snippets look like

```html
<!--#counter file=".count"-->
```

or similar in your HTML source. (But because they’re replaced with the actual numeric output of the counter in the page before it’s delivered to the browser, you can’t see this with a “view source” on the page. Try it: visit my company home page and view the source to see the counter on the bottom: <www.intuitive.com>).

This works well for static pages, but the output of the CGI program isn’t parsed by the Web server prior to its being sent to the client browser, so short of rewiring your server, it’s not a solution to this particular dilemma.

And so the solution is to have a general-purpose counter subroutine that you can drop into your CGI programs as they’re developed.

Version One: Classic UNIX File Locking

The challenge with a counter, of course, is that you need to compensate for possible race conditions where two instantiations of the program might step on each other during the open-file/read-contents/increment/save-new-contents loop. The traditional strategy is to use a separate .lock file and that’s what this first version of the subroutine does:

```c
int
visitcount ()
{
/**
 * How many times has this routine been called? Use temp file
 * COUNTER to keep track and LOCKFILE as a lock file.
 */

FILE *fd;
char buffer[40];
int current_value, lockid, loopcount = 0;
while ((lockid = open(LockFILE, O_CREAT | O_EXCL, 0777)) < 0) {
    usleep(10000);
    if (loopcount++ > MAXWAIT) {
        return(DEFAULT_VALUE);
    }
}
if ((fd = fopen(COUNTER, "r")) == NULL)
    current_value = 0;
else {
    fscanf(fd, "%d", &current_value);
    (void) fclose(fd);
}
```
current_value++; /* increment! */
if ((fd = fopen(COUNTER, "w")) != NULL) {
    fprintf(fd, "%d\n", current_value);
    (void) fclose(fd);
}
(void) close(lockid);
(void) unlink(LOCKFILE);
return(current_value);
}

Here are the relevant definitions. LOCKFILE should be set to the name of a lock file, usually on the same file device as the counter file. COUNTER is the name of the file within which the subroutine keeps track of visitor count. MAXWAITS indicates how many times the program can go into the usleep() sleep wait loop (you’ll want to keep this low if it’s a CGI program). DEFAULT_VALUE is the value to return if we can’t get the lock file.

When run on Linux 2.0.30, this counter sporadically failed and lost track of the counter value, which was highly annoying. On other versions of UNIX it was more reliable but that didn’t solve the problem within my code!

**Version Two: Flock**

The solution was to modify the code to use the flock() file-locking mechanism, which begat some modifications to the program:

```c
int visitcount()
{
    /* How many times has this routine been called? Use temp file COUNTER to keep track and LOCKFILE as a lock file.
     **/
    FILE *fd;
    char buffer[40];
    int current_value, loopcount = 0;
    fd = fopen(COUNTER, "r");
    while (flock(fileno(fd), LOCK_EX | LOCK_NB) != 0) {
        usleep(10000);
        if (loopcount++ > MAXWAITS) {
            return(-1);
        }
    }
    fscanf(fd, "%d", &current_value);
    (void) fclose(fd);
    current_value++; /* increment! */
    if ((fd = fopen(COUNTER, "w")) != NULL) {
        fprintf(fd, "%d\n", current_value);
        (void) fclose(fd);
    }
    (void) flock(fileno(fd), LOCK_UN);
    return(current_value);
}
```

In addition to being a smaller and more elegant solution, it’s also more reliable because the requirement for the atomic-level (uninterruptible) check-and-lock event is done in OS code, rather than my hoping I code it correctly in my own procedure.

I can see both of these procedures in use: the lock file strategy is demonstrated at <intuitive.com/origins>, and the flock version is shown at <www.trivial.net>.
A logical extension to this would be to allow multiple counters in the same CGI (indeed, that's exactly what Trivial.Net does; there's a "times started" counter and a "times completed" counter). No problem, make the filename a parameter to the procedure itself.

The other addition would be to allow it to output a graphical representation of the number rather than just text. This turns out to be surprisingly easy if you remember that the CGI itself is sending HTML to standard output. Instead of having the output "12," for example, it could output:

```
<img src=digit1.gif><img src=digit2.gif>
```

As long as the digits are in a well-known location (perhaps on their own server), splitting out a stream of individual digits would work fine. There is a small performance penalty you could incur getting a number of tiny graphic files rather than a single, slightly larger, multidigit graphic.

by Glen McCluskey

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1998. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

**using C++ as a better C**

This column covers some miscellaneous topics related to using C++ as a better C.

**Mixing C++ and C Code**

One of the common issues that always comes up with programming languages is how to mix code written in one language with code written in another.

For example, suppose that you're writing C++ code and wish to call C functions. A common case of this would be to access C functions that manipulate C-style strings, for example `strcmp()` or `strlen()`. So as a first try, we might say:

```cpp
extern size_t strlen(const char*);
```

and then use the function. This will work, at least at compile time, but will probably give a link error about an unresolved symbol.

The reason for the link error is that a typical C++ compiler will modify the name of a function or object ("mangle" it), for example to include information about the types of the arguments. As an example, a common scheme for mangling the function name `strlen(const char*)` would result in:

`strlen__FPCc`

There are two purposes for this mangling. One is to support function overloading. For example, the following two functions cannot both be called "f" in the object file symbol table:

```cpp
int f(int);
int f(double);
```
But suppose that overloading was not an issue, and in one compilation unit we have:

```c
extern void f(double);
```

and we use this function, and its name in the object file is just "f". And suppose that in another compilation unit the definition is found, as:

```c
void f(char*) {};
```

This will silently do the wrong thing – a double will be passed to a function requiring a char*. Mangling the names of functions eliminates this problem, because a linker error will instead be triggered. This technique goes by the name “type safe linkage.”

So to be able to call C functions, we need to disable name mangling. The way of doing this is to say:

```c
extern "C" size_t strlen(const char*);
```

or:

```c
extern "C" {
    size_t strlen(const char*);
    int strcmp(const char*, const char*);
}
```

This usage is commonly seen in header files that are used both by C and C++ programs. The extern "C" declarations are conditional based on whether C++ is being compiled instead of C.

Because name mangling is disabled with a declaration of this type, usage like:

```c
extern "C" {
    int f(int);
    int f(double);
}
```

is illegal (because both functions would have the name “f”).

Note that extern “C” declarations do not specify the details of what must be done to allow C++ and C code to be mixed. Name mangling is commonly part of the problem to be solved, but only part.

There are other issues with mixing languages that are beyond the scope of this presentation. The whole area of calling conventions, such as the order of argument passing, is a tricky one. For example, if every C++ compiler used the same mangling scheme for names, this would not necessarily result in object code that could be mixed and matched.

**Declaration Statements**

In C, when you write a function, all the declarations of local variables must appear at the top of the function or at the beginning of a block:

```c
void f()
{
    int x;
    /* ... */
    while (x) {
        int y;
        /* ... */
    }
}
```
Why are declaration statements useful? One benefit is that introducing variables with shorter lifetimes tends to reduce errors.

Each such variable has a lifetime that corresponds to the lifetime of the block it's declared in. So in this example, x is accessible throughout the whole function, and y is accessible inside the while loop.

In C++, declarations of this type are not required to appear only at the top of the function or block. They can appear wherever C++ statements are allowed:

```cpp
class A {
public:
  A(double);
};
void f()
{
  int x;
  /* ... */
  while (x) {
    /* ... */
  }
  int y;
  y = x + 5;
  /* ... */
  A aobj(12.34);
}
```

and so on. Such a construction is called a declaration statement. The lifetime of a variable declared in this way is from the point of declaration to the end of the block.

A special case is used with for statements:
```cpp
for (int i = 1; i <= 10; i++)
  ...
  /* i no longer available */
```

In this example the scope of i is the for statement. The rule about the scope of such variables has changed fairly recently as part of the ANSI standardization process, so your compiler may have different behavior.

Why are declaration statements useful? One benefit is that introducing variables with shorter lifetimes tends to reduce errors. You've probably encountered very large functions in C or C++ where a single variable declared at the top of the function is used and reused over and over for different purposes. With the C++ feature described here, you can introduce variables only when they're needed.

**Character Constants**

There are a couple of differences in the way that ANSI C and C++ treat character constants and arrays of characters. One of these has to do with the type of a character constant. For example:

```cpp
#include <stdio.h>
int main()
{
  printf("%d\n", sizeof ('x'));
  return 0;
}
```
If this program is compiled as ANSI C, then the value printed will be `sizeof(int)`, typically 2 on PCs and 4 on workstations. If the program is treated as C++, then the printed value will be `sizeof(char)`, defined by the draft ANSI/ISO standard to be 1. So the type of a `char` constant in C is `int`, whereas the type in C++ is `char`. Note that it's possible to have `sizeof(char) == sizeof(int)` for a given machine architecture, though not very likely.

Another difference is illustrated by this example:

```c
#include <stdio.h>
char buf[5] = "abcde";
int main()
{
    printf("%s\n", buf);
    return 0;
}
```

This is legal C, but invalid C++. The string literal requires a trailing \0 terminator, and there is not enough room in the character array for it. This is valid C, but you access the resulting array at your own risk. Without the terminating null character, a function like `printf()` may not work correctly, and the program may not even terminate.

**Function-style Casts**

In C and C++ (and Java), you can cast one object type to another by usage like:

```c
double d = 12.34;
int i = (int)d;
```

Casting in this way gets around type system checking. It may introduce problems such as loss of precision, but is useful in some cases.

In C++ it's possible to employ a different style of casting using a functional notation:

```c
double d = 12.34;
int i = int(d);
```

This example achieves the same end as the previous one.

The type of a cast using this notation is limited. For example, saying:

```c
unsigned long** p = unsigned long**((0);
```

is invalid, and would need to be replaced by:

```c
typedef unsigned long** T;
T p = T((0);
```

or by the old style:

```c
unsigned long** p = (unsigned long**)0;
```

Casting using functional notation is closely tied in with constructor calls. For example:

```c
class A {
public:
    A();
    A(int);
};

void f() {
    A a;
    a = A(37);
}
```
If we want to split hairs, a perhaps more appropriate technical name for this style of casting is “explicit type conversion.”

It is also possible have usage like:

```c++
void f()
{
    int i;
    i = int();
}
```

If this example used a class type with a default constructor, then the constructor would be called both for the declaration and the assignment. But for a fundamental type, a call like `int()` results in a zero value of the given type. In other words, `i` gets the value 0.

The reason for this feature is to support generality when templates are used. There may be a template such as:

```c++
template <class T> class A {
    void f()
    {
        T t = T();
    }
};
```

and it's desirable that the template work with any sort of type argument.
An Update on Standards Relevant to USENIX Members

by Nicholas M. Stoughton
USENIX Standards Liaison

To Amend or to Revise?

Formal standards, or at least those of the IEEE and ISO, have two distinct ways of being modified once published. The first is to amend that standard; strictly, this is intended for adding new material, though an amendment can also fix some problems with the original. The second method is a full-scale revision of the entire standard. In the world of POSIX, until now, we have been publishing amendments to the original POSIX.1 and POSIX.2 core standards.

These have added such facilities as support for realtime systems, threads, and sockets. There are several more amendments in progress, such as POSIX.1a (symbolic links and other extensions), POSIX.1h (services for reliable and available systems), POSIX.1j (more realtime), etc.

An amendment changes the base standard; if you ask for POSIX.1 today, you get all the approved amendments as part of that deal. This means that vendors have a real problem keeping up, even if it does take years to approve a standard. As soon as an amendment is published, their systems stop conforming to the standard because the standard changed under their feet.

A revision allows a whole new document to be issued, looking at the entire scope of the document again. A vendor can claim conformance to an old revision.

One of the results of the big "Future of POSIX" debate in Fort Lauderdale this January was a resolution that effectively prevents PASC, the Portable Applications Standards Committee of the IEEE Computer Society, from ever sponsoring another amendment project. Those that are in progress have two years to complete or to change course and publish their standard as a standalone document.

At the same time, we have been debating the commencement of the first revision of POSIX.1 since 1990. Such a revision would be allowed to include any new functionality already published in other standards (withdrawing that separate standard as a result), and would allow us to make all the amendments that have been published truly fit together seamlessly — or at least that's the theory.

All these standards are produced essentially by volunteer effort. Various companies and organizations (such as USENIX) pay individuals for time and expenses, but entirely on a voluntary basis. If nobody volunteers to work on a revision to POSIX.1, then that project will fail.

Organizations other than PASC are working in a scope that overlaps that of PASC, most notably, The Open Group (TOG) and ISO SC22/WG15. One proposal from TOG (reported on in the February 1998 jlogin:) was for TOG to take over much of the development and support of the POSIX standards. This proposal failed during the debate, but it has been agreed that the three groups do need to work more closely. An ad hoc committee has been formed to try and work out how such co-operation might work. Your views are actively encouraged on this.

Personally, I should like to see a coordinated single working group of the technical experts from all three sources and a synchronized ballot process involving all three procedures. Please send comments to me or to Roger Martin, chair of the ad hoc committee <rjmartin@eng.sun.com>.

The following Reports are published in this column:

- POSIX.1h: Services for Reliable, Available, and Serviceable Systems (SRASS)
- The Single UNIX Specification, Version 2

Our standards Report Editor, Nick Stoughton, welcomes dialogue between this column and you, the readers. Please send any comments you might have to:

<nick@usenix.org>
The other project that the SRASS Working Group is responsible for at present is POSIX.1m, Checkpoint/Restart. This work was originally balled as a part of POSIX.1a, but was felt to be too far from consensus and was holding that project back. POSIX.1m allows an application to save the entire state of the machine, the operating system, and the applications activities so that, if something goes wrong, a saved backup state can be brought online quickly. This draft has been developed further by the working group and will be entering a new ballot soon. Please contact Richard Scalzo for further details.


Andrew Jacey <a.jacey@opengroup.org> continues his series of articles based on the new Single UNIX Specification, Version 2.

The Single UNIX Specification, Version 2, includes the threads model and interfaces defined in POSIX.1c—1995 together with a number of extensions. These extensions, known as the X/Open Threads Extension, based on widely accepted existing industry practice, were developed by the Aspen Group and submitted to The Open Group’s Base Working Group (the group that develops operating system interface specifications within The Open Group). This article is a brief introduction to these extensions. It assumes a working knowledge of the threads model specified in POSIX.1c and threads programming concepts in general.

The X/Open Threads Extension is built upon the threads model and interfaces defined POSIX.1c, otherwise known as Pthreads. POSIX.1c contains much optional functionality. When POSIX.1c was incorporated into the Single UNIX Specification, Version 2, the majority of this optional functionality was made mandatory, and additional functionality, known as the Aspen threads extensions submission, was incorporated.
The Aspen Group

Over the past few years almost all UNIX system vendors have implemented some flavor of a threads package based on the POSIX.1c interfaces. Each vendor found that the POSIX.1c interfaces were not complete in solving all their threads requirements. Consequently, most vendors implemented extensions to their thread packages to meet those requirements.

Unfortunately for application developers, not all vendors implemented the exact same set of extensions. To make things worse, the same functionality was added, but used different interface names or parameter sets. In short, this resulted in proprietary threads interfaces that are not portable across implementations, yet certain applications, such as database engines, were making heavy use of these proprietary interfaces.

Fortunately, many of the threads extensions developed were general enough that they are easily supported on any UNIX system threads implementation. In late 1995, the Aspen Group formed a subgroup to standardize the interfaces and functionality of the common thread extensions that various UNIX system vendors had implemented. The threads extensions that came out of this work by the Aspen Group comprise extensions that were made for OSF DCE 1.0 as well as others by Sun, HP, and Digital. The Aspen Group handed the completed work over to X/Open in 1996 as a submission for consideration for inclusion in the next revision of the Single UNIX Specification.

The Aspen Group extended the POSIX.1c interfaces in the following areas:

- extended mutex attribute types
- read–write locks and attributes
- thread concurrency level
- thread stack guard size
- parallel I/O

The Aspen Group carefully followed the threads programming model specified in POSIX.1c when developing these extensions. As with POSIX.1c (and unlike traditional UNIX functions), all the new functions return zero if successful; otherwise an error number is returned to indicate the error.

The concept of attribute objects was introduced in POSIX.1c to allow implementations to extend the standard without changing the existing interfaces. Attribute objects were defined for threads, mutexes, and condition variables. Attribute objects are defined as implementation-dependent opaque types to aid extensibility, and functions are defined to allow attributes to be set or retrieved. The Aspen Group followed this model when adding the new type attribute of `pthread_mutexattr_t` and the new read–write lock attributes object `pthread_rwlockattr_t`.

Extended Mutex Attributes

POSIX.1c defines a mutex attributes object as an implementation-dependent opaque and specifies a number of attributes this object must have and a number of functions that manipulate these attributes.

The Single UNIX Specification, Version 2, specifies another mutex attribute called type. The type attribute allows applications to specify the behavior of mutex-locking operations in situations where the POSIX.1c behavior is undefined. The OSF DCE threads implementation, which was based on Draft 4 of POSIX.1c, specified a similar attribute, but the names of the attributes have changed somewhat from the OSF DCE threads implementation.

The Single UNIX Specification, Version 2, also extends the specification of the following POSIX.1c functions that manipulate mutexes:

```c
pthread_mutex_lock()
pthread_mutex_trylock()
pthread_mutex_unlock()
```

These take account of the new mutex attribute type and specify behavior declared undefined in POSIX.1c. How a calling thread acquires or releases a mutex now depends upon the mutex type attribute.

Read–Write Locks and Attributes

Read–write locks (also known as readers–writer locks) allow a thread to exclusively lock some shared data while updating that data or allow any number of threads to have simultaneous read–only access to the data.

Unlike a mutex, a read–write lock distinguishes between reading data and writing data. A mutex excludes all other threads. A read–write lock allows other threads access to the data, providing no thread is modifying the data. Thus, a read–write lock is less primitive than either a mutex–condition variable pair or a semaphore.

Application developers should consider using a read–write lock rather than a mutex to protect data that is frequently referenced but seldom modified. Most threads (readers) will be able to read the data without waiting and will have to block only when some other thread (a writer) is in the process of modifying the data. Conversely, a thread that wants to change the data is forced to wait until there are no readers. This type of lock is often used to facilitate parallel access to data on multiprocessor platforms or to avoid context switches on single processor platforms where multiple threads access the same data.

If a read–write lock becomes unlocked and there are multiple threads waiting to acquire the write lock, the implementation's scheduling policy determines which thread will acquire the read–write lock for writing. If there are multiple threads blocked on a read–write lock for both read locks and write locks, it is unspecified whether the readers or a writer acquire the lock first. However, for per-
formance reasons, implementations often favor writers over readers to avoid potential writer starvation.

A read–write lock object is an implementation–dependent opaque object. There are two different sorts of locks associated with a read–write lock – a read lock and a write lock.

A thread that wants to apply a read lock to the read–write lock can use either pthread_rwlock_rdlock() or pthread_rwlock_tryrdlock(). If pthread_rwlock_rdlock() is used, the thread acquires a read lock if a writer does not hold the write lock and there are no writers blocked on the write lock. If a read lock is not acquired, the calling thread blocks until it can acquire a lock. However, if pthread_rwlock_tryrdlock() is used, the function returns immediately with the error EBUSY if any thread holds a write lock or if there are blocked writers waiting for the write lock.

Similarly, a thread that wants to apply a write lock to the read–write lock can use either of two functions: pthread_rwlock_wrlock() or pthread_rwlock_trywrlock(). If pthread_rwlock_wrlock() is used, the thread acquires the write lock if no other reader or writer threads hold the read–write lock. If the write lock is not acquired, the thread blocks until it can acquire the write lock. However, if pthread_rwlock_trywrlock() is used, the function returns immediately with the error EBUSY if any thread is holding either a read or a write lock.

The pthread_rwlock_unlock() function is used to unlock a read–write lock object held by the calling thread. Results are undefined if the read–write lock is not held by the calling thread. If there are other read locks currently held on the read–write lock object, the read–write lock object shall remain in the read locked state, but without the current thread as one of its owners. If this function releases the last read lock for this read–write lock object, the read–write lock object will be put in the unlocked read state. If this function is called to release a write lock for this read–write lock object, the read–write lock object will be put in the unlocked state.

The same POSIX working group that developed POSIX.1b and POSIX.1c is currently developing the POSIX.1j draft standard, which specifies a set of extensions for realtime and threaded programming. This includes readers–writer locks that are nearly identical to the Single UNIX Specification, Version 2, read–write locks. The Aspen Group was aware of this draft standard, but felt that there was an immediate and urgent need for standardization in the area of read–write locks.

The following table maps the Single UNIX Specification, Version 2, read–write lock functions to their equivalent POSIX.1j draft 5 functions:

<table>
<thead>
<tr>
<th>SUS, V2</th>
<th>IEEE PASC P1003.1j</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_rwlock_init()</td>
<td>rlock_init()</td>
</tr>
<tr>
<td>pthread_rwlock_destroy()</td>
<td>rlock_destroy()</td>
</tr>
<tr>
<td>pthread_rwlock_rdlock()</td>
<td>rlock_rdlock()</td>
</tr>
<tr>
<td>pthread_rwlock_tryrdlock()</td>
<td>rlock_tryrdlock()</td>
</tr>
<tr>
<td>pthread_rwlock_wrlock()</td>
<td>rlock_wrlock()</td>
</tr>
<tr>
<td>pthread_rwlock_trywrlock()</td>
<td>rlock_trywrlock()</td>
</tr>
<tr>
<td>pthread_rwlock_unlock()</td>
<td>rlock_unlock()</td>
</tr>
</tbody>
</table>

The pthread_setconcurrency() function may also have an effect on implementations where the kernel mode and user mode schedulers cooperate to ensure that ready user threads are not prevented from running by other threads blocked in the kernel.

The pthread_getconcurrency() function always returns the value set by a previous call to pthread_setconcurrency().

**Thread Stack Guard Size**

DCE threads introduced the concept of a thread stack guard size. Most thread implementations add a region of protected memory to a thread's stack, commonly known as a guard region, as a safety measure to prevent stack pointer overflows in one thread from corrupting the contents of another thread's stack. The default size of the guard regions attribute is PAGESIZE bytes and is implementation–dependent.

Some application developers may wish to change the stack guard size. When an application creates a large number of threads, the extra page allocated for each stack may strain system resources. In addition to the extra page of memory, the kernel's memory manager has to keep track of the different protections on adjoining pages. When this is a problem, the application developer may request a guard size of 0 bytes to conserve system resources by eliminating stack overflow protection.

Conversely, an application that allocates large data structures such as arrays on the stack may wish to increase the default guard size in order to detect stack overflows. If a thread allocates two pages for a data array, a single guard page provides little protection against stack overflows because the thread can corrupt
adjoining memory beyond the guard page.

The Single UNIX Specification, Version 2, defines a new attribute of a thread attributes object; that is, the guardsize attribute that allows applications to specify the size of the guard region of a thread’s stack.

An implementation may round up the requested guard size to a multiple of the configurable system variable PAGESIZE. In this case, pthread_attr_getguardsize() returns the guard size specified by the previous pthread_attr_setguardsize() function call and not the rounded up value.

If an application is managing its own thread stacks using the stackaddr attribute, the guardsize attribute is ignored, and no stack overflow protection is provided. In this case, it is the responsibility of the application to manage stack overflow along with stack allocation.

Parallel I/O

Many I/O intensive applications, such as database engines, attempt to improve performance through the use of parallel I/O. However, POSIX.1 does not support parallel I/O very well because the current offset of a file is an attribute of the file descriptor.

Suppose two or more threads independently issue read requests on the same file. To read specific data from a file, a thread must first call lseek() to seek the proper offset in the file and then call read() to retrieve the required data. If more than one thread does this at the same time, the first thread may complete its seek call, but before it gets a chance to issue its read call, a second thread may complete its seek call, resulting in the first thread accessing incorrect data when it issues its read call. One workaround is to lock the file descriptor while seeking and reading or writing, but this reduces parallelism and adds overhead.

Instead, the Single UNIX Specification, Version 2, provides two functions to make seek/read and seek/write operations atomic. The file descriptor’s current offset is unchanged, thus allowing multiple read and write operations to proceed in parallel. This improves the I/O performance of threaded applications. The pread() function is used to do an atomic read of data from a file into a buffer. Conversely, the pwrite() function does an atomic write of data from a buffer to a file.

More Information


Additional information on the Single UNIX Specification can be obtained at The Open Group WWW site, <http://www.UNIX-systems.org/>.
the bookworm

Books reviewed in this column:

Whitfield Diffie & Susan Landau

Privacy on the Line

L.D. Stein

Web Security

Jeanne C. Adams, et al.

Fortran 95 Handbook

James Mohr

SCO Companion

Marly Poniatowski

HP-UX System Administration Handbook and Toolkit

James Carlson

PPP Design and Debugging

William Stallings

High-Speed Networks

Jennifer Stone Gonzalez

The 21st-Century Intranet

Annabel Z. Dodd

The Essential Guide to Telecommunications

W. Richard Stevens

UNIX Network Programming, 2nd ed., vol. 1

(Continued on page 79)

by Peter H. Salus

Kidnapped by gypsies at an early age, Peter H. Salus grew up in the mountain fastnesses of Buntania. Escaping at age 18, he became an international swindler until — at 25 — he retreated to a lamasery. He has no qualifications whatever.

<peter@pedant.com>

Boston can be very cold in the winter. The result is that I’ve looked at many more books than usual, and the variety of topics has gone up. Some of the books are very, very good.

Confidentiality

Diffie and Landau have produced one of the best books I’ve gotten to review. It concerns something of great importance: privacy. Moreover, it is tightly and well written. It pains me to have to point out that we’re all enmeshed in the politics of wiretapping and encryption. This book gets to the core of the debate concerning national security and civil liberties. There is a discussion of the functions of privacy and the dangers to society in its loss. A tip of my hat.

Although Stein’s book is also on security, it is nearly at the opposite scale from Diffie and Landau. It really is a “step-by-step reference guide.” If you’re running a Web site and you want to make it relatively safe, this is the book for you.

Fortran

Fortran was the first high-level programming language, issued by IBM towards the end of 1957. It was also the first programming language standardized by ANSI. And it was the first programming language I ever saw (in May 1958). We have travelled from Fortran 66 through Fortran 77 to Fortran 90 and now to Fortran 95. Adams and her fellow authors have done a super job in putting together this complete resources and reference to ISO/ANSI Fortran.

System Administration

There were a lot of works of interest to sysadmins this past month or two. A few were new editions, and I’ve mentioned them at the end of this column. Mohr’s book is far too padded to be useful. For example, I don’t think that “Users and System Administrators” really need 90 pages on shells, AWK, sed, and vi. But then, I might well be wrong: this isn’t a book for sysadmins at all, it’s a “Dummies” book in disguise, one you can carry in an airport without being embarrassed.

Poniatowski’s HP-UX sysadmin book is at the opposite pole. It really dives into the nitty-gritty of HP-UX administration. Moreover, the CDs are for Win95, WinNT, HP-UX, Solaris, AIX, and MPPAS (NCR). A useful book.

Networking

Carlson has done for PPP what Comer and Stevens have done for TCP/IP; elucidate a protocol (or suite of protocols) in such a way that the intelligent user can make sense of the material. This is a handy and useful book that made me realize just how much we all owe to the guys who participate in the IETF.

Stallings has turned out another of his lucid treatments of design principles and such complexities as congestion control in both TCP and ATM. Stallings’s brief “tutorial” in graph theory is exceptionally fine.
On a “smaller” scale, there are Intranets. In December I mentioned Danan & Odorica’s book (favorably). But now I’ve read Gonzalez’s opus, and it puts all the others in the shade. The author has succeeded in combining the technical, the financial, and the business aspects of networking into a relatively seamless narrative. Occasionally, I hit something irritating (at the end of section 1, Gonzalez seems to equate the Web and the Internet, for example); but who could remain irked when turning the page and encountering “Chapter 11: This is not Bill Gates’ Playground”? This is a very fine book.

At the other end of the scale is Dodd’s Essential Guide. It’s not. The nine (!) pages on telephony seem to concern themselves with PBXs alone. The discussions of T1/E1, etc., are totally insufficient, not even mentioning the ways that supervision is the root of much of the difference. (It’s also interesting to note that although Dodd mentions that UUNET is “owned by WorldCom,” she is silent on the facts that WorldCom has been in the process of acquiring MCI and that GTE has acquired BBN.)

Second Helpings
There are several books that have come out in new versions or new editions to which attention should be directed. Stevens’s Network Programming was good nearly a decade ago. It seems to have waxed so that there will now be (at least) two volumes. Volume 1, which covers sockets and xti, is really a new work, not merely an updating.

Equally good is the second edition of Arnold and Gosling. It’s a hundred pages longer than the 1996 version. Among the nearly 200 Java books I’ve seen, this is the very best.

Foster-Johnson’s new edition is one of the first to cover Tcl/Tk 8.0 in any detail. I found the book too full of screen dumps, but the CD has good stuff on it.

Winsor’s book on Solaris administration has been revised. If you’re running Solaris, you want to get it. However, if you’re involved with administration at all, you know the second edition of Nemeth, Snyder, and Hein. It had a CD-ROM packed into it. You can now “update” your old CD by purchasing their “Tools” CD. It’s got lots of really good stuff on it.

There’s also a new edition of Craig Hunt’s TCP/IP book. Another good ‘un.

Finally, there’s the volume I’m not certain how to classify. Sobell’s Hands-On Linux combines his earlier Linux book with Caldera’s release and with Netscape. I think it’s more new packaging than new information. I looked at OpenLinux Lite and went back to RedHat.

Ken Arnold & James Gosling
The Java Programming Language, 2nd ed.

E. Foster-Johnson
Graphical Applications with Tcl & Tk, 2nd ed.

Janice Winsor

Evi Nemeth, Garth Snyder & Trent Hein
Tools for UNIX System Administrators

Craig Hunt
TCP/IP Network Administration, 2nd ed.

Mark G. Sobell
Hands-On Linux
book reviews

Mark Lutz

Programming Python

Reviewed by Terry Rooker
trooker@illuminet.net>

Early in the 1990s the Python programming language inspired much enthusiasm. Here was a language that was easily ported across platforms. Even better, it allowed procedural, functional (as in Lisp), or object-oriented styles. It directly incorporated network programming features. Thus, much of the enthusiasm for Python was based on the ease with which GUI-based networked applications could be written for numerous platforms. All that was needed was a decent reference manual and training guide.

Programming Python is that long anticipated reference and guide, all rolled into a single package.

The first part of the book covers a description of the general features of the language. Most importantly, it describes some of the unusual characteristics of Python. Although this section is not a comprehensive tutorial, it does provide enough information to help prospective Python programmers orient themselves. For example, Python can be used as a scripting language, it can be compiled into standalone applications, it can be embeded into programs written in other programming languages (or other Python programs), or it can be interpreted.

The second section covers fundamental parts of the language. It slowly develops a sample application. With each iteration the author adds some variation that describes an additional feature of the language. By working through these examples the reader learns how these features add power to the program.

The final section deals with producing final applications. Much of it is focused on developing a GUI for a front end to the application. It also discusses how Python can be extended by including functions from other languages as Python modules. It describes in more detail embedding Python in other applications. It pretty much covers the hard-core features needed to develop real applications.

The book succeeds in its goal. For anyone interested in learning to program with Python, Python Programming is a great resource. It is important to understand what it is. Normally, to support a programming language, you'd like to see a tutorial or other beginner's introduction, a detailed guide to advanced features of the language, and finally a comprehensive reference to the language. Unfortunately, Python never caught on widely, so it is difficult to support publication of several different books. For years the Python community has lived with documents available with the Python distribution: a short tutorial, some general documents, and programming reference. Although these documents are not entirely satisfactory, they have served a purpose. Probably most lacking has been a detailed guide to the language.

Programming Python is that guide.

Because it serves as the detailed guide to provide programming help, the book is lacking in terms of introductory material and detailed descriptions of the built-in language functions. Although it is invaluable in terms of helping programmers build applications, it is less useful for someone interested in learning the language.

This problem is aggravated by the nature of the language itself. As I said previously, you can mix procedural, functional, and object-oriented styles. As a matter of fact, several developer friends of mine highly recommended Python specifically because it does allow a mixing of the styles. This book is no exception, with the examples showing how to mix the styles in many cases. For a developer looking for a rapid prototyping language, this may be less of a problem. In terms of software engineering principles, it is a disaster waiting to happen. I still have not decided if this is good or bad. Like most such dilemmas, it is probably a good things for expert programmers who can keep track of the differences and use each style appropriately. For the rest of us it may be more troubling. In either case the book could have pointed out some of these differences.

For a book trying to fill several niches, this single problem is not major. Overall, the presentation of material is well laid out, and the style is very readable. The name "Python" came from Monty Python's Flying Circus, and the author uses that theme in some of the presentation. Some of the chapter headings are inspired by skits and ideas from the British TV show, as are some of the programming examples. For someone who never watched the Monty Python show or saw the movies, these references can be confusing. Even though I was familiar with most of the references, I found them distracting because they sometimes appeared to be a forced attempt at being cute. Fortunately, there are few of them relative to the size of the book, and they don't distract from the overall presentation.

For anyone interested in the Python language this book is the best thing around. It may not be the best tutorial from which to learn the language, but any serious programmer will find the book invaluable. The problem is that Python may be overtaken by events in the networked software development environment. With the exception of its use as a scripting language, almost all the claims made for Python are also made for Java, and Java has a much wider base of support. So although Programming Python is an excellent book supporting an excellent language, that language may be relegated to a niche player. But if you want an alternative to Java, Python is a good choice, and this book is the perfect starting point.
James Carlson

**PPP Design and Debugging**

Reviewed Chris Kottaridis
<chrisk@bsdli.com>

This book covers low-level PPP communications with detailed discussions of data encoding and how HDLC is used to encapsulate PPP transmissions. It addresses the Link Control Protocol and Authentication Protocol options and provides a complete state diagram for the negotiation of those options. The book also addresses the different Network Control Protocols available with PPP and provides a summary of the options for each Network Control Protocol. There is a chapter on data transforming layers that discusses data encryption and data compression over PPP links. The chapter on bandwidth management discusses topics such as demand dialing, multilink PPP, callback, and active bandwidth management techniques, some of which are not yet well defined. The author also includes a very practical chapter that aids in interpreting PPP traces that should be very helpful in solving real-life field problems.

I found the book to be very thorough. In fact, on a cursory reading of the book, I often got bogged down in too much detail. However, I am sure that I will appreciate the level of detail when I find myself struggling with a specific PPP problem.

It is great to have a single location that has references to all the pertinent RFCs and a discussion of the history of how various RFCs have superseded others. But more important than that is James Carlson's focus on interoperability with existing implementations. He makes a point to identify which options and features you can expect to run into which helps keep you focused on the pertinent aspects of the RFCs. This is most evident in his mentioning of Microsoft's extensions to PPP even though they were not approved by the IETF. Basically, he puts the RFCs into perspective for you.

The primary audience would be PPP code developers, although network administrators who are willing to get into the nitty-gritty details of packet sniffing would also find it useful. Most system administrators could find it useful for background knowledge, but their focus is usually more on management of per-user configuration files, which is implementation specific and not addressed in this book.

All in all, it is a book I will be keeping close at hand when I do any kind of PPP work. It will probably be the first place I look when I have a question about PPP.

Stephen R. Covey, Roger A. Merrill, and Rebecca R. Merrill

**First Things First**

Reviewed by Kartik Subbarao
<subbarao@aurora.lf.hp.com>

Recently, I took a good course on time management called "First Things First," based on the book of the same title by Stephen Covey. It emphasizes linking time management decisions to one's personal priorities and overall goals. As I was thinking about it, I saw many connections between time management and process scheduling in an operating system. As Rob Kolstad observed in his "motd" column recently, the overhead of ineffective multitasking can be extremely crippling. I can certainly relate firsthand.

At the risk of sounding corny, I'll indulge in analogies between the concepts of First Things First and the software world. I hope it will be worth the risk and there will be some interesting connections. At the very least, there might be some humor value.

One of the things that First Things First teaches is to make an explicit distinction between the importance and urgency of a task. Covey draws a four-quadrant map relating the two attributes:

<table>
<thead>
<tr>
<th>Quadrant I:</th>
<th>Quadrant III:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urgent/Important</td>
<td>Urgent/Unimportant</td>
</tr>
<tr>
<td>Quadrant II:</td>
<td>Quadrant IV:</td>
</tr>
<tr>
<td>Not Urgent/Important</td>
<td>Not Urgent/Unimportant</td>
</tr>
</tbody>
</table>

Quadrant I tasks are both important and urgent. These are high-priority items that we just have to address right now—things like a disk crash or a rapidly approaching deadline. Our schedulers have gotten good at handling these tasks, since they are immediately important to our livelihood.

Quadrant II tasks are important, but not urgent. These are the things that we know we should do, but we can put them off because they aren't pressing—things like backups and commenting code, sometimes even things like taking a break or finding some time to relax. These are the CPU-starved processes that need better treatment from our schedulers.

Quadrant III tasks have an insidious sense of urgency to them, but are in fact unimportant. We do these things because they present themselves to us immediately, and they end up being major time sinks. The are things like replying to unimportant email messages as they arrive, responding to flame-bait news articles and downloading, compiling, and installing the latest version of a rarely used software package just because we see an announcement. These are the rogue processes that fool our schedulers into putting them on the run queue, but they have no business being there.
Quadrant IV tasks are both unimportant and not urgent. This is when we randomly surf the Web or read news for the express purpose of avoiding something else (as opposed to surfing the Web or reading news as part of a normal routine — that would be a Quadrant II activity). When our schedulers are caught up in Quadrant I crises and Quadrant III rat holes, we compensate by escaping to Quadrant IV, forking off random processes, and redundantly sweeping our mental caches.

It seems like the vanilla behavior for many of our schedulers is to service Quadrants I and III, focusing on urgency while sacrificing importance. This is somewhat akin to a first come first served approach. Compared to that, the first things first approach ensures that Quadrant II tasks have a high priority, and Quadrant III tasks have a low priority. A beneficial side effect of this is that completing Quadrant II tasks decreases the number of Quadrant I crises that have to be dealt with and in turn lessens the need for us to escape to Quadrant IV.

But how does one go about implementing this scheduling algorithm? There is no quick fix approach; we have to rewrite our vetware kernels. Before we can do that, we need to understand our own source code. We need to do some major code inspection and discover what our priorities really are, what we really think is important to us. Then we are ready to rearchitect our code to do the right things. (Covey refers to these as “true north” principles, those things we know to be correct, independent of ourselves.) While we’re at it, we can take the opportunity to unlink those encumbering behavioral scripts that we have implicitly copied from other people (mindless code reuse is not a good thing).

We also need to do real-time debugging. When we encounter an inconsistency between what we think we should be doing and what we’re actually doing, we need to be able to single-step through our code to figure out the problem. Covey calls this “exercising integrity in the moment of choice.” If we get good at self-awareness, we can consistently run with both -g and -O turned on without any decrease in performance. We can sense when our scheduler is about to switch to a Quadrant III task, like impulsively responding to an unimportant email message, and say “no” because we’ve internalized the higher priority of other things.

Just as with software, tracking down challenging bugs and making the code more concise, more elegant, and cleverer can be really satisfying. And the understanding that we gain in the process is truly enlightening. But at the same time, we need to guard against creeping featureism (stretching ourselves too thin by trying to do too many things) and overengineering (perfectionism, which ends up collapsing under its own weight).

The book discusses several other aspects of time management by putting them in the context of a bigger picture and is chock-full of real-life anecdotes. If you’re looking for a robust and flexible conceptual model for time management, I highly recommend First Things First.

Linda McCarthy

Intranet Security: Stories from the Trenches

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

Certainly there is a lot of talk about computer security these days, talk often driven by the media and entertainment industries. Even with “hacker” making its way into colloquial usage, most people have no idea what exactly these hackers are doing, how they’re doing it, and what havoc they really cause (I suspect we’re doomed to lose “hacker” to the media – it’s so hard to take the term “cracker” seriously). Unfortunately, many of our bosses or people who make budgeting decisions also have no idea. This is a good book for these people, although it has several important points for us techie types.

The book lives up to its subtitle with harrowing and lively accounts of intrusion incidents, many real, a few imagined for the sake of argument. Each incident provides the author with a framework to discuss the various sorts of human problems that lead to impaired security. A number of subjects are discussed, including the dangers of unmodified standard OS builds, educating inept or misinformed management, how departmental infighting weakens security, and the importance of understandable, workable policy.

Management that insists on having excellent security while downsizing will benefit greatly from the dose of reality this
book presents. The author also stresses the importance of training system administrators. We all know the importance of this, and perhaps someday more managers will get the idea. The horror stories in this book may get a few more moving in the right direction.

This book does not contain much technical detail. It does not have a checklist of files to investigate when you suspect you have a compromised machine, nor will it tell you how to use SATAN to check your network. What it does provide is outlines for various things: setting up security policy, responding to an incident, auditing your site's security. The importance of clear and concise communication is emphasized throughout and is one of the strongest features of the book.

*Intranet Security* suffers from a number of distracting stylistic flaws that will drive some readers away from it. If your manager was a literature major before being forced to switch to business, you may want to find another book. The text is liberally sprinkled with exclamation marks, and it's not hard to find groups of them bunched in threes at the ends of sentences. Emphasis is achieved by an equally liberal use of all caps. Finally, the author is very conscious of being a member of the elite group of trusted and competent security specialists and makes an equally impressive show of repeatedly omitting incident details to protect the people involved. This may be a natural consequence of the informal style of the book, but it sticks out and somewhat undermined my confidence in the author.

Keeping in mind the book's flaws and informal style, I recommend you buy this book for anyone involved in making network policy decisions, anyone unfamiliar with the realities of computer security (and insecurity), and managers who don't believe training is worth the time and cost.
The Dutch organizational form stichting, literally foundation, needs some explanation for non-Dutch readers. Under Dutch law a stichting is an institutional organization, authorized by law and with a nonprofit, and somewhat idealistic objective. A stichting does not have members. It has a board responsible for all of its activities, which legally must be in line with its objective and written regulations.

One of the founders of Stichting NLnet, Ted Lindgreen, has been heavily involved in maintaining and building the network from the start. As the first director of NLnet Holding, he designed and implemented the first national backbone in the Netherlands, using the infrastructure of the Dutch national railway. This was a significant accomplishment for that time.

By 1994, it became clear that commercial exploitation of the network needed to be supported by a more appropriate legal structure. To this purpose, NLnet Foundation established NLnet Holding B.V., a commercial company to provide high-quality Internet access to both professional and nonprofessional users.

NLnet Holding, started with 5 employees, has grown into the leading Dutch Internet provider, with more than 90 employees at the end of 1997. The holding has two daughters: NLnet Development and NLnet Services Amsterdam, and acted as a leading party in several joint ventures, such as InterNLnet, a quality access provider for the consumer market.

A year after NLnet Holding was founded it became clear to the board of NLnet Foundation that further growth relied heavily on international connectivity. More, and financially stronger, competitors were appearing, resulting in competition in the national market and pricing below cost. Cooperation with a strong (both financially and technically) international partner was deemed essential.

In 1997, negotiations with UUNET were finalized: during the negotiation phase, UUNET became a daughter of WorldCom. In August 1997, shares were swapped: all NLnet Holding shares were exchanged for a number of WorldCom shares.

Until then the role of NLnet Foundation was that of the shareholder of NLnet Holding B.V. As a result of this transaction NLnet Foundation became a very small shareholder in WorldCom. The foundation no longer had any significant influence in NLnet Holding B.V.

NLnet Foundation now faces a new challenge: to initiate new activities using its newly obtained financial, commercial, and governmental independence. The possibilities to be explored will be in line with the original NLnet Foundation's objective. This objective, dating from 1989, is to “stimulate electronic information exchange.” The current focus is on the independent (noncommercial) development and application of Internet technology. The NLnet Foundation plans to play its role as a stimulating, but not directive, organization supporting initiatives for network development. In the next issue of login: more definite ideas and further thoughts will be presented.
News from the USENIX PGP Key Signing Service

by Greg Rose
Greg, a member of the USENIX Board of Directors, manages the PGP key signing service for USENIX. He also runs the QUALCOMM’s Australian development office. He’s been involved with the use and development of UNIX since 1974.

<gr@qualcomm.com>

Everything You Probably Didn’t Want to Know About PGP at the Moment
Since I last wrote in detail about PGP for this magazine, lots of things have changed. One of them is that USENIX has about 20% more members than it did, so if some of you oldies can bear with me, I’m going to recap a little history and overview material before getting into new news.

There is a publicly, and internationally, available privacy program called PGP (Pretty Good Privacy). PGP uses public key cryptographic techniques to allow messages to be exchanged between people across public networks while both protecting the privacy of the contents and guaranteeing authenticity of the sender.

One of the major problems currently confronting the electronic commerce world is how to guarantee the authenticity of a transaction. Cryptographically, this is easy – just use digital signatures. In the real world, though, the answer is not so simple. How do you know that the cryptographic key you are using belongs to the entity (person, company, computer) you would like to think it belongs to? Or how do you send a secret message to someone when you are not sure that it isn’t his evil twin’s key?

One answer to the problem is to have trusted parties who introduce other parties to you. This is what the PGP documentation calls the “Web of Trust.” It is a web because each party in it can introduce other parties whom you may or may not already know. Using a telephone analogy, you would say secret things on the phone only if someone you trust had given you the telephone number, not if you had just looked it up in the phone book.

Another answer to the problem is to have Certification Authorities, forming a hierarchical structure. When you get a public key, you would also get a list of certificates. For example, J. Smith’s public key might come with a certificate from Widgets Inc. stating that he works for them. In turn, Widgets Inc. would need a certificate from someone stating that it is a Delaware corporation.

(Trust management and public key infrastructures are the subject of an upcoming USENIX workshop or stream and are hot research topics at the moment.)

USENIX has a service we run (at conferences) in which members can present identification while at the conference and subsequently have their PGP keys signed by USENIX, effectively “introducing” them to other PGP users. This service has been running for about 20 months now, and it has had some ups and downs, but generally it seems to provide a useful function. To find out more about the service, see <www.usenix.org/pgp/pgpintro.html>.

When we started doing this, Phil Zimmermann, the principal author of PGP, was under the cloud of indictment by the US Government for making PGP available in such a way that it was exported (by someone else unknown) in contravention of government regulations. Events since then have moved quickly. First, the indictment was dropped. Phil formed a company, PGP Inc., to try to recoup some of his devastating losses from previous years. In a funny sort of reverse buyout, PGP Inc. and Viacrypt, which had been marketing PGP commercially with a license to use the RSA public key encryption algorithms, merged. Recently, as PGP momentum gained, PGP Inc. and McAfee (antivirus software) merged to form Network Associates. Phil appears to have regained his losses.

When we started doing this, there was one kind of PGP (2.6 was its approximate number), which supported one kind of public key, based on the RSA (Rivest, Shamir, and Adelman) cryptosystem. Aye, there were the days. There was an effort going on to expand and extend PGP to support better user interfaces, more algorithms, a programming interface, and so on – this was going to be version 3.0. But like a lot of ambitious upgrades, it was a long time coming. In the meantime, Viacrypt had a new business product and needed a number. It couldn’t use 3.0 because the development was well known, so it used 4.0. Then, after the merger, when the 3.0 functionality largely became available, it didn’t make sense to call it 3.0, so it called it 5.0 instead. But much of it is what was projected two years ago. I’m going to say “old” and “new” a lot, and I hope you’ll understand what I mean.

(Warning and disclaimer: I’m trying to be as factual as I can while summarizing history. However, at times, my opinion will also come through, and I want to stress that it is my very own opinion, not that of USENIX, the editor, or the board of directors.)

It is to be expected that when you introduce new algorithms and data formats, there will be some compatibility issues. Most companies try to minimize them. (Don’t take me wrong. I think PGP Inc. tried to minimize them, too. Its “issues” were, perhaps, different from ours, though. The main issue was losing royalties to its biggest competitor for giving away a free product.) The new (PGP 5-based) products now support a number of algorithms, but most visibly, there is a different kind of public key based on the
Diffie-Hellman (or El Gamal, I don’t want to go into that) cryptosystem for encryption and the Digital Signature Standard for authentication. These have the major benefit that they are (now) unencumbered by intellectual property, whereas the patent for RSA doesn’t run out until 2000.

So there is now a version of PGP, that uses new “free” keys. It would seem to be in everyone’s best interests to use it. But, of course, the new keys are not understood by the old version. Here is where the complications set in, with a vengeance. The new free version of PGP can’t use the old keys either, because they aren’t “free.” Actually, they can, but only if you get the program from MIT, which has a license to give away RSA for non-commercial purposes (or if you are overseas, but I’ll come back to that). In particular, if you get a free version from PGP Inc., or anyone else, you don’t get to use old keys, at least not all the time, for some meaning of the word “time,” difficult to explain.

Another complication comes from the platform you are using. In those old days (two years ago), you had one command-line interface no matter what you were running. It was pretty hokey, so people disguised it a lot, but it was there. Now you have a new UNIX command-line interface, with two separate programs and four names for invoking them (five if you count the backward compatible one that just tells you it isn’t implemented yet) and (all the other ones) with almost completely incompatible arguments from the old one. There’s no command-line interface at all for Windows and Macs, though. Who needs one (besides me, that is?)

Another complication comes from geopolitical boundaries. The old PGP was illegally exported from the United States by someone unknown (or “some-many,” as new versions were generally exported within hours of becoming available), but it wasn’t at all illegal for someone outside the US to use the exported version. So PGP became widespread around the world. When the new version was about to be released, PGP Inc. took advantage of a loophole to export it legally.

(There’s a long story about that loophole. Phil Karn <http://people.qualcomm.com/karn> applied for an export license for the book Applied Cryptography by Bruce Schneier, which was granted, although it was already on sale around the world at the time. He then applied for an export license for the accompanying diskettes with source code, which was denied. He then started to sue the government.

When the applicable regulations were changed from International Traffic in Arms Regulations [State Department] to Export Administration Regulations [Commerce Department], published books and papers became explicitly exempted. This appears to be intended to derail Karn’s case or perhaps is an admission that it was silly in the first place. So to export PGP legally, PGP Inc. published it in book form, in a scannable font, with checksums on every page, and gave away copies that [surprise] were scanned in in Europe! See <http://www.pgp.com/> , where the “I” means “International,” not “Inc.”)

As I write this, the Windows version has just been scanned in and is available, but until now, it has been only the UNIX beta version. But the beta was incompatible with the released version in the US in a number of nontrivial ways. And the freely available versions in the US were only for the Windows and Macintosh platforms, not UNIX. For intellectual property reasons (not export laws), you cannot run the international version in the US. So there were more incompatibilities; the international versions supports both kinds of keys, but the US ones don’t unless you pay for them.

And then came Eudora. I need another disclaimer. I work for QUALCOMM, but these are not statements for, against, or on behalf of the company which gives away or sells Eudora. These are still my personal comments. Eudora is tightly integrated with PGP, using a plugin interface. When you get the free Eudora, you can get free PGP with it (but without RSA support). Alternatively, you can upgrade it to support RSA keys for $5. (Note that this is the cheapest way, in the US, to get full crypto functionality with PGP, although I don’t think you get all of the noncrypto features.) So, generally speaking, Eudora users (and there are a lot of them) can use PGP easily, but only with the new keys. It’s really easy to use. When you install the plugin, it walks you through a key, and it can communicate automatically with key servers and so on. Many of these users don’t understand the issues the USENIX Key Signing-Service was intended to address and generally can’t interact with the older, more knowledgeable PGP users anyway.

The large influx of less “sophisticated” users, less likely to go to Cypherpunks meetings or key signing parties, made us feel it was important to upgrade the USENIX PGP Key signing Service to support the new keys. This was not a trivial matter due to the aforementioned incompatibilities (and the not-mentioned but nevertheless plentiful bugs). This is a good place to apologize for the delays in getting the service back up and running. But it is done now. Either type of key can be signed; the query engine supports and returns both kinds. In addition to the RSA master and signature keys, there are both kinds of communication keys. Fingerprints for all these keys appear with the contact information for USENIX somewhere in every issue.

There is also an end in sight. PGP-MIME is now on a standards track at the IETF, and there are commercial certification authorities starting to serve PGP keys. We estimate that within a couple of years there will be no need for the USENIX PGP Key signing Service. We hope that it has been useful and will continue to be so for a while yet.
Twenty Years Ago in \login:

by Peter H. Salus

Peter H. Salus is the author of A Quarter Century of UNIX (1994) and Casting the Net (1995). He has known Lou Katz for over 40 years.

As I reported in the last issue, the April 1978 \login: (containing the program for the “meeting” to be held at Columbia University from May 24 to 27) preceded the March issue. Among the topics were:

- fun, games, educational uses
- V7
- graphics
- security
- networking
- database systems
- PWB
- small UNIXES (Unices?)
- biomedical and realtime applications
- legal, moral, organizational issues
- word processing and typesetting

Two things stand out: the many topics that are still pertinent 20 years later and the prominence of networking. It was only a few months since Mike Lesk had published the first UUCP paper; news had not yet been invented; and it would be years before the ARPANET reached 200 sites. Cutting edge. The “small” unices were mini-UNIX, LSI-11 UNIX, and (to some extent) MERT. I’ll talk more about the meeting in the next installment.

But the March issue carried a remarkable letter from “Lewis A. Law, Associate Director” of the Harvard Science Center. Dated March 31, 1978, it read:

I have prices for the PWB manuals. It was finally decided to divide them up as follows:

- PWB/UNIX User Manual (without Section 8) $9.90 ea.
- PWB/UNIX User Manual (Section 8 only) $2.20 ea.

Documents for the PWB/UNIX (without sections G & I) $8.40 ea.

Documents (Sections G & I only) $6.00 ea.

Purchase orders, including proof of possession of a valid license for PWB/UNIX, should be sent to: ...

PWB/UNIX was the Programmer’s WorkBench, an offshoot of the sixth edition (1976), which had begun as a third edition version for large software development projects. It was initiated by Evan Irvie in mid-1973 and led by Rudd Canaday. By June 1977, PWB supported “in excess of 1,000 users” – all within AT&T – according to Dick Haight and Ted Dolotta. But it (like so much else) seeped out of Murray Hill. PWB ran on DEC PDP-11/45s and 11/70s. In 1978, Haight and Dolotta remarked: “A typical PWB/UNIX system costs about $120,000 and can support 24 simultaneous users with ease.”

Lew’s undertaking of reproduction and distribution of the UNIX manuals meant that they would be more widely proliferated. Lou Katz told me: “Up until that time, one got Xeroxied copies from Ken.” The manuals also initiated the publishing program of the not-yet-named USENIX Association, which subsequently published the 4.1, 4.2, and 4.3BSD manuals and co-published the 4.4BSD manuals with O’Reilly & Associates.

The publication of the PWB/UNIX manuals was important in several ways. As Lou related: “It regularized the manuals. … Before that event, there really wasn’t any easy way for more than the few who actually had hands-on on the machines to get them. Once they were purchasable. …”

It was Lew Law who negotiated with AT&T to get permission to reproduce Thompson’s copies. Of course, 20 years ago, all UNIX users had individual licenses. By requiring proof of license, Harvard (and later, USENIX) covered any accusation of separating the manuals from the fully licensed software.

(As late as 1986, this Association was still requiring copies of licenses before shipping 4.2BSD manuals. A few years ago, there were still several file cabinets full of such papers.)

One wonders whether AT&T’s lawyers realized what would happen when the manuals became available. Looking at Berkeley, it was by puzzling his way through these manuals that Bill Joy got 32/V running on the (brand-new) VAX. The paging system of Babaoglu and Ferrari was based on them. And the next year we had 3BSD – a complete, bootable system. Would we have had alternate UNIX systems without the manuals? I’m sure we would have. But it would have taken longer.
K-12 Outreach: MVHS and the Student Network

by Matt Shiblea

Matt Shiblea is a network administrator for the Maryland Virtual High School and spends a significant amount of time training students and teachers in system administration issues. He currently resides in Silver Spring, MD.

Schools normally don’t look a gift horse in the mouth. With the presidential technology education initiative and a host of other programs like it, many schools have seen a recent influx of computers and computer equipment. The problem is that many schools lack the resources (both financial and personnel) to integrate and maintain this new equipment effectively. The Maryland Virtual High School (MVHS) and its subsidiary, the Student Network Administration Project (SNAP), use students to meet this need.

For years, MVHS has provided its members with computer equipment and Internet connectivity. We’ve also shown our members how effective and valuable student sysadmins can be. In most of our schools, the students become the primary resource for computer technology maintenance. In almost all cases, the students are the primary sysadmins for our Linux Internet servers.

With an eye toward improving the overall quality of these students’ experiences and toward sharing our knowledge with other schools, SNAP has begun to develop a two-part curriculum for use in secondary education. The first semester is an introductory course in computer networking. The second course introduces UNIX system administration, with some time spent using UNIX as a platform for network management. Recognizing that many schools don’t have the teacher resources to offer such advanced technology education, SNAP is also developing teacher training resources to assist new instructors of this curriculum.

Taking an early interest in the MVHS and SNAP initiatives, SAGE and USENIX invited me to submit a grant proposal. Following a face-to-face meeting during LISA ’97, the USENIX Board of Directors voted to fund 50% of the SNAP proposal over the course of the next three years. This funding would permit MVHS to develop, test, revise, and publish the SNAP curricula and training materials.

It is our hope that schools around the nation and in other parts of the world will benefit from these materials and the documentation of our efforts. To accomplish our goals, we are using a three-stage model of development. The materials are first developed by SNAP. They are written to a draft phase and then tested in the classroom by the SNAP coordinator. Next, the materials are disseminated to supporting teachers who have many years of experience in computer science and secondary school instruction. These teachers further test the materials and make recommendations for improvements. The third phase involves a larger dissemination to participating teachers. These teachers also test the materials and provide feedback for their improvement. The materials will then be published along with a record of the trials and the improvements made as a result.

Development for the Networking I curriculum began in the summer of 1997. The first trial of that curriculum took place at Montgomery Blair High School in Montgomery County, MD during the fall semester of the 1997-1998 academic year. Two further trials of the Networking I curriculum are under way during the second semester of the ’97-’98 school year, one at Northern High School in Garrett County, MD, and another at James M. Bennett High School in Wicomico County, MD. We are in the process of revising the curriculum based on these early trials and of preparing the classroom notes and materials for electronic dissemination. We expect to expand the curriculum review process with trials at new locations during the 1998-1999 academic year. Work has begun on developing the teacher training materials for Networking I and a first version is expected by the end of the 1997-1998 academic year. We expect trials of the Networking II/Systems Administration curriculum to begin in the second semester of the 1998-1999 academic year, with additional trials to follow.

SNAP has a way to go before it is completed, but the initial efforts are promising. The first course is a significant step in helping students build their interest, knowledge, and skills in an advanced area of computer science. These students then have the ability to help their schools and communities meet the growing need for technical assistance with computer systems. Helping others to troubleshoot problems gives SNAP students an opportunity to enhance their understanding of technical issues through hands-on experience in a real-world environment.

You can find out more about the Maryland Virtual High School at <http://mvhs1.mchs.edu/mvhs.html>, You can find out more about the Student Network Administration Project at <http://mvhs1.mchs.edu/snap/index.html>.
SPECIAL OFFER FOR USENIX MEMBERS ONLY!

Usenix and the IEEE Computer Society announce a new agreement that benefits you! IEEE Concurrency is now available to Usenix members through technical cosponsorship at a 40% discount. And, if you subscribe now, you get a bonus issue absolutely free!

IEEE Concurrency covers advances in computer architectures and programming languages that enable you to achieve new levels of system performance by distributing functions across networks and allowing different processors to operate concurrently on multiple aspects of a problem. Departments, tutorials, and feature articles in IEEE Concurrency give software and hardware perspectives on subjects as diverse as collaborative computing, real-time systems, database and transaction processing, heterogeneous computing, and visualization.

IEEE Concurrency—practical information you need to stay on the cutting edge.

Departments and columns keep you current

Industry Spotlight
Global Broadcast (world news and developments)
Focus (application features)
New Products

competitive

Mobile Computing Trends
Object-Oriented Track
Distributed Databases
Multimedia Applications
Distributed Internet Computing

informed

Viewpoint
Interview
Virtual Roundtable
Calendar
Book Reviews

Feature articles focus on topics like:

Actors and Agents
Distributed Databases
FPGA Computing
Mobile Computing
OO Technology
And more!

FREE BONUS ISSUE

Order now and get the April–June issue focusing on FPGA Computing, free!

Don't delay—order today!

RISK FREE USENIX MEMBER Order Form

☐ YES, SIGN ME UP!
YES! I want to subscribe to a full year of IEEE Concurrency at the special Usenix rate of only $49 and receive a free bonus issue on FPGA Computing!

Useenix individual member ID# ________________________________
(Required for this offer)

Name: ____________________________

MAILING ADDRESS:

City ____________________________

STATE/COUNTRY ZIP/Postal code ____________________________

☐ Check enclosed

☐ Charge to: ☐ Visa ☐ MasterCard ☐ American Express

CHARGE-CARD NUMBER: ____________________________

EXPIRATION DATE: ____________________________

☐ USA, PLEASE INCLUDE BILLING ADDRESS ZIP CODE

SIGNATURE: ____________________________

All prices are in US dollars; payment must accompany order. Residents of DC, FL, and Canada, please add applicable sales tax.

TO ORDER, send complete information and payment to:

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314
phone 1-714-821-8380
Fax for fastest service to 1-714-821-4641

100% MONEY-BACK GUARANTEE—— If you are ever dissatisfied, you may cancel your subscription and receive 100% of your money back.
# 1998 USENIX Annual Technical Conference

**June 15-19, 1998**  
**Marriott Hotel**  
**New Orleans, Louisiana**

## Tutorial Program

Each tutorial runs from 9:00 AM to 5:00 PM. Sorry, no partial or split-day registrations are allowed.

<table>
<thead>
<tr>
<th><strong>Monday, June 15, 1998</strong></th>
<th><strong>Tuesday, June 16, 1998</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M1</strong> System and Network Performance Tuning, <em>Hal Stern</em></td>
<td><strong>T1</strong> Solaris Internals <em>New, Marc Staveley</em></td>
</tr>
<tr>
<td><strong>M2</strong> Classic Topics in System Administration, <em>Trent Hein &amp; Eva Nemeth</em></td>
<td><strong>T2</strong> Hot Topics in Modern System Administration <em>New, Trent Hein &amp; Eva Nemeth</em></td>
</tr>
<tr>
<td><strong>M3</strong> Inside the Linux Kernel <em>Updated, Stephen C. Tweedie</em></td>
<td><strong>T3</strong> Linux Systems Administration <em>New, Bryan C. Andregg</em></td>
</tr>
<tr>
<td><strong>M5</strong> Real World Applications of Cryptography <em>New, Greg Rose</em></td>
<td><strong>T5</strong> Secure Communications Over Open Networks <em>New, Marcus J. Ranum</em></td>
</tr>
<tr>
<td><strong>M7</strong> Introduction to Java <em>New, Prithvi Rao</em></td>
<td><strong>T7</strong> CGI and WWW Programming in Perl, <em>Tom Christiansen</em></td>
</tr>
<tr>
<td><strong>M8</strong> Introduction to Perl for Programmers, <em>Tom Christiansen</em></td>
<td><strong>T8</strong> Network Security Profiles: What Every Hacker Already Knows About You and What To Do About It, <em>Jon Rochlis and Brad Johnson</em></td>
</tr>
<tr>
<td><strong>M9</strong> Internet Security for UNIX System Administrators, <em>Ed DeHart</em></td>
<td><strong>T9</strong> Windows NT Security <em>New, Rik Farrow</em></td>
</tr>
<tr>
<td><strong>M10</strong> Security Around the World Wide Web <em>Updated, Daniel Geer &amp; Jon Rochlis</em></td>
<td><strong>T10</strong> Sendmail Configuration and Operation <em>New, Eric Allman</em></td>
</tr>
<tr>
<td><strong>M11</strong> Troubleshooting Firewalls <em>New, Char Sample</em></td>
<td><strong>T11</strong> Web and Intranet Performance: A Quantitative Analysis <em>New, Daniel Menascé &amp; Virgilio Almeida</em></td>
</tr>
</tbody>
</table>

---

**REGISTER EARLY FOR TUTORIALS AND GET YOUR FIRST CHOICE**
1998 USENIX Annual Technical Conference

Technical Sessions  Wednesday, June 17, 1998

Joint Opening Session
Opening Remarks and Awards
Fred Douglass, AT&T Labs – Research

Keynote Address: Science and the Chimera
James “The Amazing” Randi

Refereed Papers

Performance I
Scalable Kernel Performance for Internet Servers Under Realistic Loads
Gaurav Banga, Rice University and Jeffrey C. Mogul, Digital Equipment Corporation, Western Research Lab

Tribeca: A System for Managing Large Databases of Network Traffic
Mark Sullivan, Juno Online Services and Andrew Heybey, Niksun

Transparent Result Caching
Amin Vehdat, University of California, Berkeley and Thomas E. Anderson University of Washington

Extensibility
SLIC: An Extensibility System for Commodity Operating Systems
Douglas P. Shirmley, University of California, Berkeley; Steven H. Rodrigues, Network Appliance Corporation; David Petrou, Carnegie Mellon University; and Thomas E. Anderson, University of Washington

A Transactional Memory in an Extensible Operating System
Yasushi Saito and Brian Bershad, University of Washington

Dynamic C++ Classes - A Lightweight Mechanism to Update Code in a Running Program
Gisle Hjälmtysson, AT&T Labs - Research and Robert Gray, Dartmouth College

Commercial Applications
Fast Consistency Checking for the Solaris File System
Kent Peacock, Ashvink Kamaraju, and Sanjay Agrawal, Sun Microsystems

General Purpose Operating System Support for Multiple Page Sizes
Narayan Ganapathy and Curt Schimmel, Silicon Graphics, Inc.

Implementation of Multiple Pagesize Support in HP-UX
Indira Subramanian, Cliff Mather, Kurt Peterson, and Balakrishna Raghunath, Hewlett-Packard Company

FREENIX Track

Concurrent Session
Maintaining Common Drivers
Matt Thomas, Internet Locksmith

A Machine-Independent DMA Framework for Net BSD
Jason R. Thorpe, NASA Ames Research Center

Concurrent Session
Panel Discussion: Whither IPSec
Moderator: A. D. Keromytis, University of Pennsylvania
Panellists: John Ioannidis, AT&T Labs – Research; Theodore Tso, MIT; Hugh Daniel, Linux FreeS/WAN Project
Others to be announced

Concurrent Session
NetBSD Operating System

Concurrent Session
Host ATM Research Platform (HARP)
Timothy J. Salo, Network Computing Services

Dummynet and Forward Error Correction
Luigi Rizzo, Università di Pisa

Concurrent Session
OpenBSD Operating System
Theo de Raadt, The OpenBSD Project

Concurrent Session
Arka: Freely Available AFS Client
Johan Danielsson, Royal Institute of Technology; Assar Westerlund, Swedish Institute of Computer Science

Portable NTFS Driver
Martin V. Loewis, Humboldt University, Berlin

Invited Talks

Repetitive Strain Injury (RSI): Causes, Treatment, and Prevention
Jeff Okamoto, Hewlett-Packard

Mixing UNIX and PC Operating Systems via Microkernels: Experiences Using Rhapsody for Apple Environments and OpenNT for NT Systems
Stephen R. Walli, Safeway Systems and Brett Halle, Apple Computer

Succumbing to the Dark Side of the Force: The Internet as seen from an Adult Web Site
Daniel Klein, Erotika
1998 USENIX Annual Technical Conference

Technical Sessions    Thursday, June 18, 1998

Joint Session: Historical UNIX

Reflections on the '73 CACM Paper
Dennis Ritchie, Lucent Technologies, Bell Laboratories

20th Anniversary of the First Port of UNIX
Steve Johnson, Transmeta; Richard Millar, Miller Research; and Juris Reinfelds, New Mexico State University

Refereed Papers

Performance II
SimICS/Sun4m: A Virtual Workstation
Peter S. Magnusson, Fredrik Larsson, Andreas Moestedt, Bengt Werner, Swedish Institute of Computer Science; Jim Nilsson, Per Stenström, Fredrik Lundholm, Magnus Karlsson, Fredrik Dahlgron, Dept. of Computer Engineering, Chalmers Univ. of Technology; Håkan Grahn, Dept. of Computer Science, Univ. of Karlskrona/Ronneby

High-Performance Caching With The Lava Hit-Server
Jochen Liedtke, Vsevolod Panteleenko, Trent Jaeger, and Nayeem Islam, IBM

Cheating the I/O Bottleneck: Network Storage with Trapeze/Myrinet
Darrell C. Anderson, Jeffrey S. Chase, Syam Gaddie, Andrew J. Gallatin, and Kenneth G. Yocum, Duke University; Michael J. Feeley, University of British Columbia

Neat Stuff
Mhz: Anatomy of a Micro-benchmark
Carl Staelin, Hewlett-Packard Laboratories and Larry McCoy, McCoy, Inc.

Automatic Program Transformation with JOIE
Geoff A. Cohen and Jeffrey S. Chase, Duke University; David L. Kaminsky, IBM

Deducing Similarities in Java Sources from Bytecodes
Brenda S. Baker, Bell Laboratories and Udi Manber, University of Arizona

Work-In-Progress Reports (WIPs)
Submission deadline: May 1, 1998
Submissions to: wips98@usenix.org

The WIPs session will consist of five-minute presentations. If you have work-in-progress, we invite you to submit a 1 or 2 page abstract via email in plain text to: wips98@usenix.org by May 1. Please include your name, affiliation, and the title of your talk.

FREENIX Track

Concurrent Session
Design and Implementation of a SCSI Subsystem

Multimedia Driver Support
James Lowe, University of Wisconsin, Milwaukee

Concurrent Session
ISC DHCP Distribution
Ted Lemon, Internet Software Consortium

Heimdal: 116N Free Kerberos Implementation
Johan Danielsson, Royal Institute of Technology; Assar Westerlund, Swedish Institute of Computer Science

Invited Talks

Software Development Models: The Cathedral and The Bazaar
Marshall Kirk McKusick, Author and Consultant, and Eric S. Raymond

Real Programmers Don't Always Use C
Henry Spencer, SP Systems

ADAPT: A Flexible Solution for Managing the DNS
Jim Reid and Anton Holleman, Origin b.v.

Software Development Models: The Cathedral and The Bazaar
Marshall Kirk McKusick, Author and Consultant, and Eric S. Raymond

Real Programmers Don't Always Use C
Henry Spencer, SP Systems

ADAPT: A Flexible Solution for Managing the DNS
Jim Reid and Anton Holleman, Origin b.v.

Linux Operating System
To Be Announced

FREENIX BoF
The Free Software Foundation: Projects and Futures
Richard Stallman, The Free Software Foundation

FREENIX BoF
Licensing
Jon "maddog" Hall, Linux International

REGISTER BY MAY 8 AND SAVE UP TO $100
### Refereed Papers

**Networking**
- Transformer Tunnels: A Framework for Providing Route Specific Adaptations  
  Pradeep Sudama and B. R. Badrinath, Rutgers University
- The Design and Implementation of an IPv6/IPv4 Network Address and Protocol Translator  
  Marc E. Fiaczynski, Vincent K. Lam, and Brian N. Barshed, University of Washington
- Increasing Effective Link Bandwidth by Suppressing Replicated Data  
  Jonathan R. Santos and David J. Wetherall, MIT

**Real Time**
- Making Commodity PCs Fit for Signal Processing  
  Michael Isner, MIT
- The Eclipse Operating System: Providing Quality of Service via Reservation Domains  
  John Bruno, Eran Gaab, Banu Özden, and Avi Silberschatz, Lucent Technologies, Bell Labs
- A Framework for Alternate Queueing: Towards Traffic Management by PC-UNIX Based Routers  
  Kenjiro Cho, Sony Computer Science Laboratory, Inc.

**Security**
- Implementing Multiple Protection Domains in Java  
  Chris Hawbital, Chi-Chao Chang, Grzegorz Czajkowski, Deyu Hu, and Thorsten von Eicken, Cornell University
- The Safe-Tcl Security Model  
  Jacob Y. Levy, Laurent Demaillay, John Ousterhout, and Brent Welch, Sun Microsystems Laboratories, Inc.

### FREEUNIX Track

**Concurrent Session**
- malloc(3) Revisited  
  Poul-Henning Kamp, The FreeBSD Project
- Kernel Sched/ZOUNDS  
  Ron Minnich, Sarnoff Corporation

**Concurrent Session**
- imail - Fidonet  
  Eugene Crosser, Sovam Teleport
- Samba as WNT Domain Controller  
  John Blair, University of Alabama

**Concurrent Session**
- K Desktop Environment  
  B. J. Wuebben, Cornell University
- GNOME Desktop Project  
  Miguel de Icaza and Federico Meno, Universidad Nacional Autónoma de México; Elliot Lee, Columbia Union College; Tom Tromey, Cygnus Solutions

**Concurrent Session**
- Console Server  
  Banson Matheson, Ferguson Enterprises
- Linux Emulation for SCO  
  Ron Record, Santa Cruz Operation

**Concurrent Session**
- Kawala—Compiling Dynamic Languages to Java VM  
  Per Bothner, Cygnus Solutions
- Samba Futures  
  Jeremy Allison, Whistle Communications

**Concurrent Session**
- User API for Tape Drives  
  Odysseas I. Pentakalos and Aram Khalili, University of Maryland

### Invited Talks

**Highlights from USENIX Conferences & Symposia**

**Panel Discussion: Is a Clustered Computer In Your Future?**
- Panelists to be Announced

**The Future of the Internet**
- John S. Quarterman, Matrix Information and Directory Services (MIDS)

### Joint Closing Session

**Beyond Wearable Computing: Personal Imaging as Example of Humanistic Intelligence**
- Steve Mann, University of Toronto

---

**Updates:** [http://www.usenix.org/events/no98/](http://www.usenix.org/events/no98/)
SANS Conference
THE SEVENTH ANNUAL SYSTEM ADMINISTRATION,
NETWORKING AND SECURITY CONFERENCE
The DoubleTree Hotel and Monterey Convention Center • Monterey, CA • May 9-15, 1998

SATURDAY

Sa-1 System and Network Performance Tuning
Hal Stem’s in-depth session on how to measure and optimize individual systems and networks of systems.

Sa-2 Advanced Windows NT Security
Gene Schulz takes you beyond the basics of NT Security.

Sa-3/ Su-3 Basic Perl Programming (2-day course)
Learn from the master Tom Christiansen—in a unique environment where you’ll bring your own laptop for in-class exercises.

Sa-4 UNIX Security: Threats and Solutions
The foundation course in UNIX security by Matt Bishop, the nation’s most respected teacher of security.

Sa-5 Building a Successful Security Infrastructure
Michele Crabbe’s classic course provides a step-by-step guide to creating a successful infrastructure and bridging the gap between technology and management.

Sa-6 Firewall Management and Troubleshooting
One of the most experienced firewall implementers, Char Sample, shares techniques that allow you to fix your firewall problems without compromising security.

Sa-7 An Introduction to Tc and Tk Programming
Mark Meretzky helps you understand the more confusing components of Tc and shows you how to build graphical interfaces using Tk.

Sa-8 Securing Solaris: Step-By-Step
Hal Pomeranz provides a step-by-step program for building a bastion host with Solaris. Top rated!

SUNDAY ALL DAY

Su-1 DNS and Sendmail for the Enterprise: Step-By-Step
Hal Pomeranz provides another practical, step-by-step course introducing the concepts needed to handle mail routing and namespace administration.

Su-2 Designing Predictable Distributed Systems
Hal Stern and Evan Marcus show you how to gain control over distributed systems.

Su-3/ Su-3 Basic Perl Programming (2-day course)
Christiansen (continued)

SUNDAY AM

Am-1 UNIX Network Security
Where are the vulnerabilities in your UNIX network? Matt Bishop shows you the holes and how they can be plugged.

Am-2 Incident Response: Scenarios and Tactics
Our field’s most popular story-teller, Randy Marchany, provides real-world case studies on how to respond to more than a half-dozen types of security incidents.

Am-3 Fundamentals of IPv6
Kevin Lahey shows how Version 6 solves many problems of IPv4.

Am-4 Expect Programming
The expert segment in the Tc/Tk program—from Mark Meretzky—Sa-7 is a pre-requisite.

Am-5 Managing the Transition from sendmail to qmail
Russell Nelson shows you how to fix sendmail problems by replacing sendmail with qmail.

SUNDAY AFTERNOON

Pm-1 UNIX Security: Writing Secure Programs
Matt Bishop helps you make sure you don’t add new security vulnerabilities when you develop Sethid programs.

Pm-2 SSH Introduction to Implementation
Steve Acheson shows how to take advantage of this powerful tool for secure remote access.

Pm-3 Introduction to the IP Security Protocols (IPSec)
Ran Atkinson shows you what is involved in this important new security standard and how it will be applied.

Pm-4 TCP/IP Troubleshooting with UNIX
A step by step approach, what to look for, what it should look like, what to do about it by Jim Hiekle.

Pm-5 Network Address Translation
John Stewart shows you how to use this technique for managing growing IP spaces and partners across the internet.
MONDAY

10-1 UNIX Security Tools: Use and Comparison
Matt Bishop's most popular course covering the public domain security tools and how to make them work for you.

10-2 CGI and WWW Programming in PERL
Dan Klein shows you how to use Perl to give your web sites, more functions, better interactivity.

10-3 Security on the Web
Dave Kenski and John Stewart provide a live, interactive class on securing web sites.

10-4 Administering Sendmail in the Real World
SANS Executive Chair, Bob Kolstad, offers lessons learned from 15 years of sendmail experience and handling more than 3 million messages per day.

10-5 Oracle Database Management for System Administrators
Unique information on user management, database expansion, security and more, all based on real-world experience. By Scottie Swenson.

10-6 Introduction to UNIX System Administration
Just the basics, so you know where to focus, from Peter Galvin.

10-7 Introduction to Networking and TCP/IP
Steve Acheson provides the language and technology fundamentals to help sysadmins cope with networking issues.

10-8 Intrusion Detection Using Traffic Analysis
Learn how to monitor your systems to identify suspect patterns and events—from our highest rated new speaker, Steve Nohbcutt.

TUESDAY & WEDNESDAY

CONFERENCE—DAY ONE
Tuesday Short Courses

Virtual Private Networks in the Real World
Tina Bird, Cerner Corporation

DNS Security: Secure Naming and Key Distribution
Donald Eastlake, III, Cybercash, Inc.

Managing Your PartnerNets
Michele D. Crabb, Cisco Systems, Inc.

Design and Implementation of Highly Available Systems
Andrew Rieger, Lehman Brothers and Phil Brandenberger, Lehman Brothers

Firewall Architectures and Product Selection
Char Sample, Firewall and Security Consultant

Building Client Relationships—How To Stop Managing Your Users
Andrew Rieger, Lehman Brothers

CONFERENCE—DAY TWO
Wednesday Short Courses

Network Based Denial Of Service Attacks: Trends, Descriptions, and How to Protect Your Network
Craig A. Huegen, Cisco Systems, Inc.

Remote Access Authentication and Authorization Technologies—An Overview
Michele D. Crabb, Cisco Systems, Inc.

Help Desk Techniques
Laura LeHew, Deer Run Associates

Oracle and UNIX Performance Tuning
Ahmed Alomari, Oracle

Effective Use of PGP—Pretty Good Privacy
Mitch Baker, Nichols Research Corporation

FRIDAY

Fr-1 Advanced Heterogeneous Systems Management—UNIX and NT
Real-world, tested techniques, proven on Wall Street, for integrating UNIX and Windows NT systems, taught by Dr. Yuval Liro and Andrew Reiger.

Fr-2 Advanced Network Engineering and Capacity Planning
Jake Hartinger provides a fast-paced guide to both LAN and WAN capacity planning tools and techniques.

Fr-3 Building Internet Firewalls
No theory here. Just practical guidance from a top firewall guru, Marcus Ranum.

Fr-4 Securing the Network with Kerberos
Dan Geer and John Rochlis offer a valuable insider's view of this increasingly important technology.

Th-5/5 Jumpstart JAVA (2-day course)
Fr-5 continued

REGISTER
Visit our website at www.sans.org
Questions? 719-599-4303 or email SANS98@sans.org
Computer-Abetted Education

I admit it. When it comes to education, I am somewhat of an elitist. I really think that a good education can help. I also think that a poor education is part of the problem, not part of the solution. There are plenty of places to obtain both, of course.

Computer Aided Education has been a hot buzzword for how long, 25 years? The PLATO project at the University of Illinois was at the forefront for a long time. Fabulous stuff. If you needed drill and practice, it was the absolute bees’ knees. If you needed many people to use a clever simulation, it was great.

Two of the most popular “lessons” (programs) were a chemistry lab distillation experiment (complete with sound: Boom! Bang! Tinkle!) that enabled you to blow up a lab safely, and a simulation of a teacher's first year in school. Different principles require different behaviors, and your job was to retain your job (or even get tenure). It was interesting to use the clues given to try to classify the principle and then react to the requests correctly.

PCs, the Web, and cheap computers have brought us back again to computer aided education (since the programs are not too difficult to write and can be lucrative, I reckon). I do not know that the lessons of PLATO have been heeded. While the drill and practice lessons and the simulations were successful, straight presentation of material (“page-turners”) was not. In fact, outside of its strong points, I believe PLATO was consistently identified as the weak link in the teaching chain.

Enter the Web, the CD-ROM encyclopedia, and word processors for the home. Nowadays, if you’re a fifth grader, you need only pull up your Encarta entry for the topic du jour and mouse it over into the word processor. Bring over a few illustrations and voila! you have a fabulous, illustrated (probably in color!) essay. Of course, the skills being cultivated were different probably from the ones the teacher hoped.

Big deal, you might say. Students have been copying essays from the encyclopedia forever. In fact, I recall that my principal means of creating elementary school essays was the encyclopedia. I would hand copy text, paraphrasing and reorganizing as I went. At least I processed the words, if only by reading and then writing them.

It’s even easier now! Consider the following email to Jeff Polk, <polk@delos.com> (delos has mythological origins; I’m sure that’s how the address came to be used):

Dear Jim,

I would appreciate if you could possibly inform me where I can obtain information on Icarus. This is a 8th grade school project and there is really not alot of information on this person. The most we get is just small paragraphs and it has to be at least three (3) pages. It would be greatly appreciated if you could be of help.

Thank you for your time and will be waiting to hear from you.

Sincerely, Sue

I looked at Alta Vista. Sue’s right; there’s not much. Poor Sue is going to have to go all the way over to the library and then ask a librarian to learn that Icarus is a popular star of ancient mythology. Of course, it’s much easier to stay in your chair and send email to someone else asking them to get the work done for you. Good training: wrong skill.

I don’t have a prescription here other than to exhort all of you to ensure that your children or others whom you mentor do not end up in situations like this! Someone has to be at the end of the “buck stops here” train and this is surely not the way for it to happen.
wish is the "windowing shell" for Tcl/Tk applications. It may be the development solution that you've been looking for.

With Tcl/Tk, you can create graphical user interfaces in short order. You can write one program, and it will run cross-platform with a native look-and-feel on UNIX, Windows 95/NT and Macintosh systems. You can add Java beans or integrate your own customized C code, so you never hit the "wall" that you find in other packages. You can embed your programs in a Web page to create interactive content for your Web site. And you can do all of this for free—with no purchase price, no royalties, and no licensing agreements.

So why isn't everyone doing this? Well, they are. Sybase uses more than 1,000,000 lines of Tcl code to perform regression testing on their database product. Shell has an oil rig in the Gulf of Mexico that's controlled by Tcl/Tk. Pixar uses Tcl/Tk to coordinate the animation of computer-generated characters. SCO uses Tcl/Tk to build the administration tools for their UNIX products. Web sites such as the Java Beans Directory, Java Solutions Online, and the Apple Developers Catalog Online are all powered by Tcl. The list of uses goes on and on.

It's easy to get started with Tcl/Tk. Just visit our Web site, and we'll show you how.

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

WEB SITE
http://www.usenix.org

AUTOMATIC INFORMATION SERVER
If you do not have access to the Web, finger <info@usenix.org> and you will be directed to the catalog which outlines all conferences, activities, and services.

PGP INFORMATION
All correspondence to:
1998 Operational Key
Key ID: 1024/F6F82613 1997/11/21
USENIX 1998 <pgp@usenix.org>
Key fingerprint = 80 6f 85 48 C2 IA 88 45 48 5F F2 28 E6 41 B0 F1

1998 Signing Key
Key ID: 1024/4F75A901 1997/11/21
USENIX 1998 Signature
<http:l/www.usenix.org/pgp/pgpsig.html>
Key fingerprint = 05 FD CF A1 21 F4 70 00 5C 69 C2 25 E4 66 89 A6 9B
USENIX master key <not-for-email>:
Key ID: 1024/2FEA2EF1 1996/04/08
Key fingerprint = DB A7 50 99 66 E4 E4 A9 80 B2 D9 E2 FE DA 00 5E

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

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of ;login: are June 9, 1998 and August 11, 1998.

ADVERTISING ACCEPTED
;login: offers an exceptional opportunity to reach 9,000 leading technical professionals worldwide.

Contact: Cynthia Deno
cynthia@usenix.org
408 335 9445

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

POSTMASTER
Send Address Changes to ;login:
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES