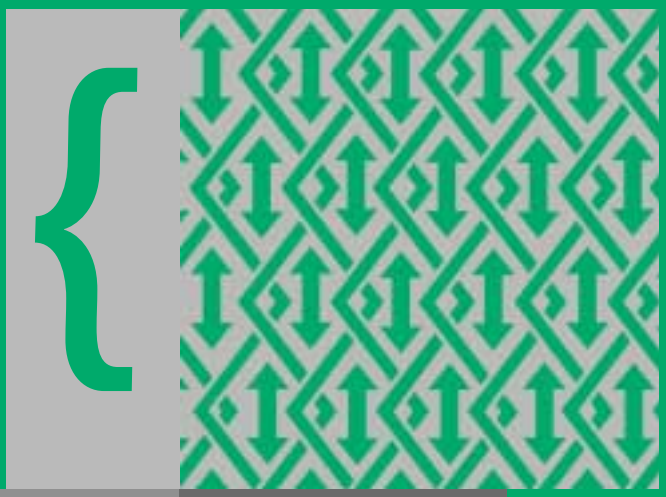
;login:

THE MAGAZINE OF USENIX & SAGE

October 2000 • volume 25 • number 6



inside:

MOTD

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

motd

by Rob Kolstad

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

<kolstad@usenix.org>

Professionalism

I enjoyed the special privilege of seeing a concert by the 70s and 80s band Earth, Wind, and Fire recently. They were preceded by a very competent blues combo, The Hazel Miller Blues Band, a perfectly reasonable group performing blues vocals with keyboards and guitars.

But when EW&F took the stage, the difference was not only immediately apparent but incredibly dramatic. It was clear that these guys knew what they were doing (probably proving that practicing your instruments for 20 years really does lead to a certain level of proficiency). Their music and performance had impact and verve – quite astounding (no different from any other super-competent band that one might like, in general, of course).

I mused to myself about the notion of professionalism closer to the technical world. I just reviewed/edited a very good perl book about debugging that's soon to be published. It included all sorts of advice on how to be a programmer whose attributes I might term "professional."

One of the book's most amusing suggestions, to me, was a simple methodology for taking trouble reports. It required just three questions:

"What did you observe?"
"What did you do to cause that?"
"What did you expect to happen?"

These questions can be answered by people at any technical level from total novice/layman all the way to the top of the guru heap. Furthermore, as a technical person, you're liable to get 90% or more of the information you really need in order to solve a technical-style problem.

I tried to think on how many times people have asked those questions of me when I called a help desk or vendor support telephone line. I'm not sure that I have been treated professionally in those contexts in the last half decade or so.

Programmers can also exhibit a degree of professionalism. Coding that doesn't contain buffer over-runs is one way to be more professional. Following advice found in Kernighan's and Pike's books is also a good way to move up the ladder (e.g., "verify all input before acting on it").

Writing such things must be hard as not enough programs exhibit such properties (e.g., feeding binary garbage to UNIX utilities used to cause problems). Completing web-based forms is often a nightmare, as is trying to get the formatting (e.g., credit cards) in just the format the programmer had in mind.

The SAGE folks have a new set of ethics. Dan Geer's column on open systems is provocative. I am going to continue to reflect on the challenge of putting on a professional face both for peers and for the outside world to see. It seems like quite a challenge.

apropos

Living on Borrowed Time

by Tina Darmohray

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

<tmd@usenix.org>

System administrators like to tell "war stories." I hypothesize that our retelling of stories serves our community much the same way story-telling serves many cultures: by passing along traditions, values, and experience. With the explosive growth of our industry, it seems there are ever more experiences to share and more stories to tell. I received this note from one of my friends via email last week.

I'm writing this note from the co-location cage at Exodus. Good connectivity from here :-) Over the last few months that I've been in here, I've wondered who the next cage over belongs to. I've never seen such a mess. Their racks are stuffed full of bare mother-boards (no cases) that are as crooked as the hind leg of a dog, bowed in the middle from the weight of the CPUs and other components. There's a "bridge" of sorts over the doorway. It is built up of those long power strips (plugged in and in use, of course) on each side. Over the top, connecting the two sides, are two Panduit wire-management trough covers that are duct-taped together. They support a thick bundle of Ethernet cables. The right side support is held up by being duct-taped to a shelf. The left side support is wedged between two stacks of computers piled on the floor, each about waist-high. On top of the pile closest to the door is an HP hub. The place makes your belly churn to look at it. I've always wondered who this mess belonged to. Today I found out. It's [Editor's Note: name removed to protect the innocent] . . .

Stories like this do make seasoned system administrators wonder why that hardware is in such disarray. The popular excuse these days is often "the company is running on Internet time." I gotta say, this one doesn't hold much water for me any more because in fact, it's often the very reason the company's network should be a showplace for industry best practices. It's obvious, isn't it? If a company's business is the network, doesn't it follow that the network is the most important priority and has to be done right? Sadly, as we hear from my friend, it's not always the case, which is what makes it noteworthy.

Cultures tell stories that have special meaning and have the intent to pass on wisdom to the listener. Veteran system administrators are virtually screaming at the tops of their lungs in many forums to take the time to design and deploy networks that can scale, be reliable, and are secure. Heed their advice and avoid the temptation to cut corners and create network monstrosities in the name of "Internet Time." If you do, you'll find out all too soon that you're living on "Borrowed Time."

;login:

EDITORIAL STAFF

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

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

MANAGING EDITOR
Jane-Ellen Long <*jel@usenix.org*>

COPY EDITOR
Eileen Cohen

TYPESETTER
Festina Lente

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

letters to the editor

HACKER VERSUS CRACKER

FROM MATT CURTIN
<*cmcurtin@interhack.net*>

I'd like to add my support to Joseph E. Wals's request that USENIX be more conscientious in its use of "hacker" vs. "cracker." I think John Walker, founder of AutoDesk, most clearly articulated why when he wrote in *The Hacker's Diet*:

> The word "hacker" and the culture it connotes is too rich to sacrifice on the altar of the evening news.

To that let me add only that it should be we who influence the mainstream media for the better, not the reverse.

Rob replies:

If only we had enough clout or numbers to influence the mainstream media. Regrettably, with the majority of our USA citizens (and a totality of its journalists) with access to computers, I fear we have no hope of winning this battle by leading the charge.

A BURGESS FAN

FROM: MARKO SCHUETZ
<*marko@kinetic.ki.informatik.uni-frankfurt.de*>

I spoke my mind in the past, when I wasn't pleased with *;login:* content. This time I'd like to let you know, that I thoroughly enjoyed reading "System Administration Research" parts 1 & 2. Encouraging research in this direction and in the accessible way the articles do sure has my backing!

A DEBT OF GRATITUDE

FROM TED DOLOTTA

Regarding the article "Teaching Operating Systems with Source Code UNIX" by Bob Gray in the July 2000 *;login:* (vol. 25, no. 4), we should recall the original – and, in the context of the history of UNIX, very, very significant – use of UNIX to teach operating-system design, namely the operating-systems courses taught by the late John Lions at the University of New South Wales in the 1970s.

John Lions wrote, for these courses, two companion volumes (*UNIX Operating System Source Code – Level Six* and *A Commentary on the UNIX Operating System*), which together constitute the first complete explanation of the UNIX operating system, apart from being very well-written and a model of pedagogical clarity.

Bob replies:

We all owe a debt of gratitude to the late John Lions who taught Operating Systems based on Source Code UNIX in the late 1970s. His material has been republished so that we may now grow from it.

conference reports

USENIX Annual Technical Conference
June 18–23, 2000
San Diego, CA

Opening Session

Summary by Josh Simon

Announcements

Christopher Small announced that there were 1,730 attendees as of the start of the technical sessions. The program committee received 91 refereed-paper submissions (up from 63 in 1999) and accepted 27 (up from 23 in 1999). Next year USENIX will be in Boston.

Christopher Small

The best paper awards were:

Best paper: "Scalable Content-aware Request Distribution in Cluster-based Network Servers" by Mohit Aron, Darren Sanders, Peter Druschel, and Willy Zwaenepoel

Best student papers (tie): "Integrating a Command Shell into a Web Browser" by Robert C. Miller and Brad A. Myers, and "Virtual Services: A New Abstraction for Server Consolidation" by John Reumann, Ashish Mehra, Kang G. Shin, and Dilip Kandlur

Kirk McKusick, chair of the Freenix committee, spoke about that track. The committee received 56 refereed paper submissions and accepted 29 of them. Most of the papers were open source–related. Because the Freenix track was run with the same rigor as the general refereed papers track, they also presented awards:

Best paper: "An Operating System in Java for the Lego Mindstorms RCS Microcontroller" by Pekka Nikander

Best student paper: "Protocol Independence Using the Sockets API" by Craig Metz

Andrew Hume, immediate past president of the USENIX Association (and now vice president) announced the two annual USENIX awards:

Software Tools User Group (STUG) Award: Tetu Yionen for the initial design and creation of "ssh", the Secure Shell.

The Flame: Outstanding Achievement Award given posthumously to W. Richard Stevens for his work as an innovator, teacher, and active member of the USENIX community (accepted on his behalf by his sister, his widow, and his children)

Keynote Address

Bill Joy presented the keynote address on his vision on the future of technology.

Based on his 25 years of experience, Joy forecasted the next 25-30 years in computing. He started by looking back at the history: the eventual acceptance of software as research in computer science, the integration of networks and the operating system, and the growth of portability in computing. More and more we'll see standards defined in English, perhaps passing code or agents instead. He also sees the continued need for maintaining compatibility with open protocols and specifications, noting that protocols often outlive the hardware systems for which they were designed.

Looking forward, Joy believes that Moore's Law will continue. He expects to see up to a 10^{12} improvement over 30 years based in part on molecular elec-

This issue's report is on the USENIX Annual Technical Conference held in San Diego, California, June 18-23, 2000.

Thanks to the summarizers:

Doug Fales
Rik Farrow
Kevin E. Fu
Matt Grapenthien
Bob Gray
Josh Kelley
Jeff Schouten
Josh Simon
Craig Soules
Gustavo Vegas

And check out Peter Salus' impressions of the conference on page 34.

A collection of photographs taken at the conference can be found at *<http://www.usenix.org/publications/library/proceedings/usenix2000/photos.html>*

<http://www.usenix.org/publications/library/proceedings/usenix2000/photos.html>

tronics and improved algorithms. The question of synchronization between different geographies becomes hard when you can store 64 TB in a device the size of your ballpoint pen. We need to improve resilience and autonomy for the administration of these devices to be possible.

Further, he sees six webs of organization of content: near, the Web of today, used from a desktop; far, entertainment, used from your couch; here, the devices on you, like pagers and cell phones and PDAs; and weird, such as voice-based computing for tasks like driving your car and asking for directions. These four would be the user-visible webs; the remaining would be e-business, for business-to-business computing, and pervasive computing, such as Java and XML.

Finally, reliability and scalability will become even more important. Not only will we need hardware fault tolerance but also software fault tolerance. In addition we need to work toward a distributed consensus model so there's no one system in charge of a decision in case that system is damaged. This leads into the concepts of byzantine fault tolerance and the genetic diversity of modular code. We also need to look into the fault tolerance of the user; for example, have the computer assist the user who has forgotten her password.

Refereed Papers Track

SESSION: INSTRUMENTATION AND VISUALIZATION

Summarized by Josh Simon

Mapping and Visualizing the Internet

Hal Burch, Carnegie Mellon University; Bill Cheswick and Steve Branigan, Bell Labs Research, Lucent Technologies

We need tools to be able to map networks of an arbitrarily large size, for tomography and topography. This work is intended to complement the work of CAIDA. So Cheswick, et al. developed tools UNIX-style, using C and shell scripts to map the Internet as well as the Lucent intranet. The tools scan up to 500 networks at once and are throttled down to 100 packets per second. This generates 100–200MB of text data (which compresses to 5–10MB) per day and covers on the order of 120,000 nodes. See <http://www.cs.bell-labs.com/who/ches/map/> for details and maps.

Measuring and Characterizing System Behavior Using Kernel-Level Event Logging

Karim Yaghmour and Michel R. Dagenais, Ecole Polytechnique de Montréal

Karim Yaghmour spoke on the problem of visualizing system behavior. `ps` and `top` are good, but neither provides truly realtime data. He therefore developed a kernel trace facility with a daemon that logs to a file. He instrumented the Linux kernel to trace the events, and then performs offline analysis of the data. The tools do not add much overhead for server-side operations but do add a lot of overhead to intensive applications such as the Common Desktop Environment (CDE). Data is collected up to 500kb per second but it compresses well. Future work includes quality-of-service kernels (throttling the rate of, for example, file opens), security auditing, and even integrating the event facility further into the kernel. Sources are available at *<http://www.opersys.com/LTT>* and are under the GNU Public License.

Pandora: A Flexible Network Monitoring Platform

Simon Patarin and Mesaac Makpangou, INRIA SOR Group, Rocquencourt

Patarin and Makpangou's goal was to produce a flexible network-monitoring platform with online processing, good performance, and no impact on the environment. The privacy of users was also important in the design. They decided to use components for flexibility and a stack model. They developed a small configuration language and a dispatcher that coordinates the creation and destruction of the components. The tool is 15,000 lines of C++, using `libpcap`. The overhead is about 0.26 microseconds per filter per packet. For example, HTTP requests get over 75Mb/s throughput on traces, which translates into 44–88Mb/s in real-world situations, or 600–2600 requests per second. Future work includes improving the performance and flexibility. More details are available from *<http://www-sor.inria.fr/projects/relais>* and released pursuant to the GPL.

INVITED TALKS

Computer System Security: Is There Really a Threat?

Avi Rubin, AT&T Research

Summarized By Rik Farrow

Avi Rubin began his talk by alluding to the February 2000 distributed denial-of-service (DDoS) attacks, saying that those attacks just about made this talk redundant. But perhaps the theme of his talk was that not enough has been learned about mitigating the threats to security, that lessons have not been learned from the past.

Rubin used the Internet Worm to illustrate this point. In November of 1988, Robert T. Morris released the Worm, a relatively small piece of code that used the Internet to copy itself to over 6,000 systems. At the time, those 6,000 systems compromised a large percentage of hosts on the Internet. Damage estimates fell between $10,000 and $97 million. Recovering from the Worm was difficult, as most Internet-connected sites relied on email for communication, and email couldn't get through while Worms were executing.

The Worm used three mechanisms to obtain access to networked systems: a buffer-overflow in the finger server, a backdoor still in sendmail (debug), and the "r" commands. The Worm would

crack passwords, using a list of 432 "common" passwords so it could assume other user identities for use with the remote shell.

Today, the finger server has been patched and the debug backdoor commented out (except for short period when a Sun engineer accidentally reenabled it in 1994) of sendmail. But "r" commands are still in use today, and SSH when used as an "r" command replacement (with trust) is still vulnerable to Worm-style attacks.

But the biggest problem then and now was homogeneity. The Worm targeted Sun servers. (A few VMS systems were also affected.) Microsoft systems present the current homogenous environment.

Rubin went on to talk about several recent virus attacks. Rubin had firsthand experience with HAPPY99, as it infected his mom's system. HAPPY99 is very polite and very widespread. It keeps a list of infected systems, can be commanded to remove itself, and does nothing else. It will even replace the DLL (Dynamic Linked Library) it modifies with a copy of the original.

The Melissa virus caused many millions in damages. Named after a dancer in Florida, Melissa first appeared on alt.sex. Because Melissa is written in Visual Basic, it is easy to modify even if you don't know VB, as you can just change its message or various strings, and it will still work (and have a different anti-virus signature as well).

Rubin listed the reasons why Melissa was so successful (besides its promise of a list of hot porn sites):

- Many people use the same mailer (Outlook)
- MSWord on Windows on almost all systems
- Many people click okay to macro warnings
- No separation between MSWord and OS
- Virus protect only works against known viruses

"If every time you received a phone call, there was a chance that an appliance, such as your toaster, could explode, you would likely not put up with it. For some reason, people are willing to tolerate the potential loss of all their data when they receive an email." said Rubin.

There have been other interesting viruses, such as Russian New Year and Bubbleboy. Russian New Year invokes Excel to execute any program on the system. Bubbleboy was the first email virus that did not require an attachment, but did display a dialog prompting the user to delete update.hta instead of doing it itself via code, which it could have done. Bubbleboy represents the killer transport mechanism, as it could install any software wanted. Actually, Microsoft makes this easy, as there are system calls that upload files (one call), then execute them (second call).

In summary, Rubin suggested that there were certain deficiencies in our current security model. Besides the homogeneity of desktops, Rubin said, "This seems to be the theme, security by pop-up windows. Do you want to continue? The default is YES." He also suggested the use of Digest Authentication in HTTP to prevent replay attacks, and said that SSL was the right place to do encryption, as the top of the IP stack, where you could tell it is working.

Rubin stated, "Trinoo [a DDoS tool] requires password (people can be very proprietary about their daemons)." He went on to recommend that people remember to make backups, so if something does go awry, you at least can recover.

There were a number of questions. Chris Harrison, Whitehead Institute, asked: "Why haven't we yet seen someone who has caused some really serious damage on a national scale?" Rubin answered, "I don't have an answer to that. Could have been much much worse. We need a sociologist to answer that."

Some one asked about the patch to Outlook. Rubin said that he would install it, but what surprises him is how fast MS produced a patch (440k).

Some one asked two questions. How much risk do you see from polymorphic viruses? How much risk from cross-platform viruses? Rubin answered, "I think polymorphic viruses are harder to detect, making them more dangerous. As for cross platform, right now all viruses and worms are on MS platforms, but when cell phones start including command interpreters, this problem will spread this as well."

Dan Geer, now CTO of @stake (which bought the l0pht) asked: What is your opinion about closing security holes by advertising them? Rubin answered by saying that there are "days I wake up feeling we should advertise vulnerabilities, and other days where I think this is a bad idea." The NIPC (National Infrastructure Protection Council, [I think]) is planning on creating industry forums for exchanging security information and rumors. Rubin said, "We need a way to disseminate information to all of the good guys and none of the bad guys." (laugh)

There were several more questions, including comments about Linux or UNIX users having the inboxes crammed full of impotent email viruses, and a suggestion to use litigation to force better adherence to security practices.

FREENIX TRACK

SESSION: STORAGE SYSTEMS

Summarized by Kevin E. Fu

Swarm: A Log-Structured Storage System for Linux

Ian Murdock and John H. Hartman, University of Arizona

Ian Murdock described Swarm, a log-structured interface to a cluster of storage devices in Linux. Filesystem interfaces are often tightly coupled to the interface of the underlying storage system (e.g., requiring a block-oriented storage device). Moving away from this tendency, Swarm provides a configurable and extensible infrastructure that may be used to build high-level storage abstractions and functionality. The intent was to build more than a just a research prototype.

Swarm is storage system, not a filesystem. However, Murdock presented two filesystems that build upon Swarm. Sting is a log-structured filesystem similar to Sprite's LFS. Ext2fs/Swarm is an unmodified Ext2 filesystem on top of a special logical Swarm device.

Several goals guided the design of Swarm:

- Scalability. As the network grows, scale to demands.
- High performance. Take advantage of the network-disk bottleneck.
- High Availability. Handle failures gracefully.
- Cost-performance. Run on commodity hardware.
- Flexibility.

Swarm moves the high-level abstractions from the server to the client. The server becomes a simple repository for data while clients implement the bulk of functionality. By clustering devices, Swarm removes centralization. In this way, Swarm avoids bottlenecks and single points of failure.

Clients use a striped, append-only log abstraction to store data on servers. This combines log-structured properties with RAID. Each client maintains its own log to eliminate synchronization. The log allows reconstruction of data from server crashes and client failures.

Logs are divided into fixed-size pieces called fragments, which are striped across storage servers. Parity is computed for redundancy. The log is append-only, conceptually infinite, and consists of an ordered stream of blocks and records.

Visit <*http://www.cs.arizona.edu/swarm/*> for more information.

Q: What is the performance to availability tradeoff? A: Write performance is good because Swarm can batch small writes. Large write performance is also good. Read performance is not as good as it should be, but this is an artifact of the implementation. The log structure gives both high availability and high performance.

Q: Is there a mechanism for synchronization of a shared resource? A: We are working on distributed lock service.

Q: How does the cleaner know what is cleanable? A: The cleaner is not aware of what is in the fragments, but it can know what parts of the fragment are alive. Agents participate in the cleaning process.

Q: You might consider a Hierarchical Storage Management (HSM) design instead of cleaning the tapes. A: Cleaning is a very interesting topic we are exploring. Your suggestion might be useful.

DMFS - A Data Migration File System for NetBSD

William Studenmund, Veridian MRJ Technology Solutions

Bill Studenmund talked about the Data Migration File System (DMFS) designed at the NASA Ames Research Center.

Implemented for NetBSD, DMFS stores files on both disk and tape. This allows an administrator to transparently archive data to tape without affecting users.

NAStore 3 is a storage system at NASA's Numerical Aerospace Simulation facility. It consists of three main systems: the volume-management system for tape robotics, the virtual volume-management system to avoid small writes to tape by aggregating data, and the data-migration filesystem. Studenmund focused on DMFS for the remainder of the talk.

DMFS is a layered filesystem that places its migration code between kernel accesses and the underlying filesystem. The system works with the Berkeley Fast File System (FFS), FFS with soft updates, and any other filesystem with a fixed number of inodes. The layering decouples development of DMFS from that of any FFS-related development.

To have control over archiving and restoring, there are a number of new FNCTL operations, such as set-archive-in-progress and set-restore-in-progress. Also, a modified `ls` command denotes whether a file is unarchived, archived and resident, or archived and nonresident.

When a user asks for a file stored on tape, a DMFS daemon blocks the user process until the file is sufficiently restored from tape or the restoration fails. However, a user can still interrupt the process by hitting control-C. There is also a utility to force restoration of files from tape into the resident risk. Writes also block until restoration finishes. The archive subsystem stores whole files on tape. Therefore, it is necessary to read from tape to piece together the unmodified parts of a file. To initiate an archival process, one can either run a user utility or let the daemon schedule an archive.

The performance cost of using DMFS is minimal. There is an overhead of 1–2% to access resident files. However,

Studenmund jokingly admitted that for obvious reasons `vi` takes about five minutes to start when the data is retrieved from tape.

Q: Looking at HSM, it seems that some files are active, while others are accessed only once a month. Is there a way to mark file as readable but not actually perform the restoration? Say, classify files as active or read once a month? A: This is not supported by DMFS, but sounds reasonably easy to add to the system.

Q: You added system calls to deal with opening/closing files. Why not use a new filesystem with vnodes instead? A: We added only three system calls. Data passed to open is sent to the name-lookup system before the filesystem gets the request. Using a special filesystem that would take file handles as the "names" of files would not work. open(2), stat(2), and statfs(2) take null-terminated strings to name files. File handles have an embedded length, and can contain nuls. Thus these system calls will not read in the correct amount of data for such a special filesystem to be able to work.

Q: Can you mark files as preferred online rather than archival? A: Everything is done in response to the migration daemon. You could modify the daemon to control this. It is certainly feasible.

A 3-Tier RAID Storage System with RAID1, RAID5, and Compressed RAID5 for Linux

K. Gopinath, Nitin Muppalaneni, N. Suresh Kumar, and Pankaj Risbood, Indian Institute of Science, Bangalore

K. Gopinath discussed the design and implementation of a host-based driver for a 3-tier RAID storage system. The tiers include a small RAID1, a larger RAID5, and a compressed RAID5. The system migrates the most frequently accessed data to the RAID1 tier. The project was motivated by the need for fast, reliable, efficient, quickly recoverable, and easily manageable storage.

Gopinath and his students previously developed the system for Solaris, but are now developing in the context of Linux. The team ran into difficulty with the Linux device-driver framework. Linux has some support to logically integrate multiple disks as one logical disk, but this breaks its own device-driver framework.

Q: A lot of RAID performance is guided by stripe size. How do you support changing stripe size? A: We do not, but we considered issues of the filesystem informing the lower layer (volume manager) about stripe sizes and in the reverse direction about the actual capacities available because of compression.

SESSION: FILE SYSTEMS

Summarized by Kevin E. Fu

A Comparison of File System Workloads

Drew Roselli and Jacob R. Lorch, University of California at Berkeley; Thomas E. Anderson, University of Washington

Drew Roselli presented an analysis of filesystem traces from a variety of environments under modern workloads. The goal was to understand how modern workloads affect the performance of filesystems.

Traces were collected on four workloads: an HP-UX instructional workload with 20 hosts for eight months (INS), an HP-UX research workload with 14 hosts for one year (RES), an HP-UX Web database and server with one host for six weeks (WEB), and a personal computing workload with eight NT hosts for four to nine months (NT). To normalize the measurements, only 31 days of each trace were used. None of the workloads reached a steady state because more files were created than deleted. Drawing chuckles from the crowd, Drew noted that "disk sales confirm this result."

The authors developed a new metric for measuring lifetimes. This involves tracking all files created during the trace and ignoring files created toward the end of the trace window. This measurement reveals that for many workloads, the block lifetime is less than the 30-second write delay used by many systems.

On to the results. Each workload has its own unique shape. Systems with daemons tend to have a knee because of activities such as overwriting Web-browser cache files. Other operating systems tend to either delete immediately or wait until space fills (e.g., emptying the recycle bin). The deletion sweet spot centers around one hour on all but the NT workload. The NT workload reads and writes more than twice the amount of the RES or INS workload. Common to all workloads are that overwrites cause the most significant fraction of deleted blocks, and overwrites show substantial locality. The results also show that a small cache is very effective for decreasing disk reads.

Because many processes memory-map a small set of files, these files are usually cached. The authors also reconfirmed a bimodal distribution where many files are repeatedly written without being read while many are repeatedly read without being written. This buys predictability in post-cache access patterns.

The UNIX traces are publicly available on *<http://tracehost.cs.berkeley.edu/traces.html>*.

Q: How often should such trace studies be done? Can we subscribe to something that collects and publishes such traces? A: If you're thinking of collecting traces, don't do it. Read others' traces instead. Workloads change over time.

Q: Did you filter out access to shared libraries? Or is this included in your trace? A: There isn't much left if you take

out the shared libraries, so we kept them in the study.

Q: Are there traces of large sequential flows? This is important to manufacturers. I'm willing to pay for the traces. A: Usually traces do not exist because people are concerned about privacy. [Margo Seltzer from Harvard stood up and said she can help in this area. Contact her offline.]

Q: Are filesystems generally designed with database workloads in mind? A: The Web server has a database in it. IBM compared database workloads with Sprite. There is not a big difference between their results and Sprite.

Q: You mentioned that your tested filesystem is not in steady state. How much is expanding? A: We reported this in paper submission, but we thought it was too wordy so we removed it. I can't remember the growth off the top of my head, but it was somewhere around 30GB.

FiST: A Language for Stackable File Systems

Erez Zadok and Jason Nieh, Columbia University

Erez Zadok discussed FiST, a language for stackable filesystems. Building a filesystem is hard! The kernel is a hostile environment; portability is a pain; development is time-consuming; and maintaining the code is costly. A stackable filesystem makes code extensible and modular. To create a new filesystem, one does not have to remember all the low-level details.

Early research suggested creating new APIs such as the stackable vnodes. However, this suffers from several problems including: modifying each operating system, rewriting existing filesystems, taking a performance hit, using a non-portable API, and having no high-level functionality. A stackable filesystem alone is not enough. It still leaves much kernel work to do.

The FiST approach consists of a language and a set of templates. The language contains simple, high-level primitives while C templates provide kernel-level abstractions. The fistgen program outputs code given a base template (basefs) and a FiST input file. Erez dubbed this system "a YACC for filesystems."

Without stacking, a user system call is translated into a filesystem call. With stacking, there is a translation between the filesystem and basefs. The translation can modify results and arguments. This allows for feature additions such as file attributes and distributed filesystems.

The authors compared the development of FiST-based filesystems with filesystems written from scratch. Four filesystems were implemented under the various models:

- Snoopfs: warns of possible unauthorized access
- Cryptfs: transparent encryption filesystem
- Aclfs: adds ACLs to files
- Unionfs: joins content of two directories

The measured systems include filesystems written from scratch in C, written using the basefs templates, written using the wrapfs templates, and written using FiST.

The authors found that the size of newly written code is one to two orders of magnitude smaller when written with FiST. Development time is shortened by a factor of seven compared to writing from scratch. Meanwhile, the performance overhead ranges between 0.8% and 2.1% to enable stacking.

For more information, see <*http://www.cs.columbia.edu/~ezk/research/fist/*> or email <*fist@cs.columbia.edu*>.

Q: Could you explain the difference in cryptfs results between basefs and wrapfs? Why the strange spike in the graph? A: Wrapfs performs unnecessary memory copies.

Q: How do you handle locking? A: Filesystem templates take care of nasty details like locking and reference counting. We use whatever the VFS provides. Special locking primitives could be added to the FiST language.

Q: What is your intention now that you have built a filesystem framework? Do you intend to write an original filesystem using FiST? A: We have plans to extend the language for low-level filesystems on various storage media.

Q: How do you handle traditional filesystem issues such as consistency and recovery? A: Stacking is completely independent of what happens above and below the stacking. You don't have to know about details.

Journaling Versus Soft Updates: Asynchronous Metadata Protection in File Systems

Margo I. Seltzer, Harvard University; Gregory R. Ganger, Carnegie Mellon University; M. Kirk McKusick, Author & Consultant; Keith A. Smith, Harvard University; Craig A. N. Soules, Carnegie Mellon University; Christopher A. Stein, Harvard University

Chris Stein compared two technologies for improving the performance of metadata operations and recovery: journaling and soft updates. Journaling uses an auxiliary log to record metadata operations while soft updates uses ordered writes to ensure metadata consistency.

Stein discussed three properties for reliability:

- Integrity. Metadata always recoverable.
- Durability. Persistence of metadata.
- Atomicity. No partial metadata visible after recovery.

The metadata update problem concerns proper ordering of operations such that a filesystem can recover from a crash.

There are three general approaches: synchronous writes, soft updates, and journaling. The Berkeley Fast File System (FFS) uses synchronous writes to maintain consistency. That is, an operation blocks until metadata are safely written to disk. Synchronous writes significantly impair the ability of a filesystem to achieve high performance. On the other hand, soft updates and journaling use asynchronous writes.

Soft updates uses delayed metadata writes to improve performance. However, delayed writes alone could change the order of writes. Therefore, soft updates maintains dependency information to order writes. That is, before a delayed write is written to disk, it must satisfy certain dependency constraints. Soft updates is not durable or atomic; has weak guarantees on when updates flush to disk; and requires no recovery process after a crash.

Journaling writes all metadata updates to a log in such a way to guarantee recoverability. It can perform asynchronous with Write Ahead Logging (WAL) to ensure that the metadata log is written to disk before any associated data. After a crash, the system can scan this log to recover metadata consistency. This provides atomicity. With synchronous journaling, the filesystem will have durability. Asynchronous mode results in higher performance at the cost of durability.

The authors performed a microbenchmark consisting of several stages reminiscent of the LFS benchmark. After creating a directory hierarchy, the benchmark writes, reads, and deletes either 128MB of data or 512 files, whichever generates the most files. There is also a test on zero-length files to stress metadata operations.

Macrobenchmarks included four workloads:

- An SSH build (similar to the Modified Andrew Benchmark)
- A netnews server (large working set, no locality)
- The SDET software development environment (timesharing)
- The PostMark ISP benchmark (small file transactions to simulate email and e-commerce)

These benchmarks were applied to two journaling filesystems; a soft updates filesystem, and FFS. One of these journaling implementations has the characteristic that the log is a separate filesystem. This makes it easy to run either synchronously or asynchronously, switching on and off the filesystem's metadata-durability property.

The results show that both journaling and soft updates dramatically improve the performance of metadata operations. However, synchronous journaling alone is insufficient to solve the metadata update problem. Synchronous journaling can penalize performance up to 90% relative to the upper bound of an asynchronous filesystem with no protection. The delayed writes in soft updates improves performance in the microbenchmarks. However, soft updates did not achieve good performance in the netnews benchmark because of the ordering constraints. On the other hand, the delayed writes let soft updates excel in performance on the PostMark benchmark.

By understanding the solutions in terms of the transactional properties they offer, one can derive the relative costs of the properties. The cost of durability is the performance difference between the best-performing solution that does offer this property and the best-performing one that does not. Likewise with the other properties. What one discovers from comparing performance is that the cost of durability is generally large relative to that of integrity. If one sacrifices durability, performance can be very close to that of systems with no metadata protection, while maintaining integrity.

Q: What part of the performance gain is due to the hefty hardware you're using? Which has a better gain, an expensive disk or an expensive CPU?
A: Synchronous systems would perform better with better SCSI drives and rotation. Asynchronous systems would benefit from more memory and faster CPU.

Q: Did you turn off disk caching? For instance, Cheetah hard drives do caching. A: I can't recall off the top of my head.

Q: In your microbenchmark, what were the file sizes and how long short lived were the files? A: There were many file sizes. There are four phases in the microbenchmark. All files were first created, the filesystem was unmounted then remounted, and then the files were read and deleted.

INVITED TALK

Watching the Waist of the Protocol Hourglass

Steve Deering, Cisco Systems

Summarized by Rik Farrow

Steve Deering elected to use imagery and puns to get across his point: that the waist of IP should remain narrow. The basic image was an hourglass shape, where layers four through seven of IP (application and transport) represent the top half; layer three (IP) is the narrow middle, or waist; and various different transport mechanisms fit into the fat bottom half (ATM, SONET, Ethernet, etc.). Deering pointed out the importance of the IP layer – that it isolates the stuff above it from the stuff below it, and that this is what has allowed the Internet to evolve.

IP allows us to create a "virtualized network" to isolate end-to-end protocols from network details/changes. Having a single IP protocol maximizes interoperability and minimizes the service interface. IP means that we can assume a least

common denominator of network functionality to maximize the number of networks.

Then Deering asked rhetorically: why worry about the waist of IP? He gave mostly humorous answers:

- It provides for navel gazing.
- It happens at middle age.
- The IP is the only layer I can get my arms around.
- I am worried about how the architecture is now being lost: the waste.
- This theme offers all these puns.

Deering went on to describe some problems with the waist of IP. For example, IP multicast (which he actually had a lot to do with) and quality of service (QoS) have added to the waist. And a mid-life crises, IPv6, has created a second waist (the hourglass image now has two narrow connections). IPv6 doubles the numbers of interfaces, requires a new Ethertype, and changes in application software.

The reason for IPv6 was that the IPv4 address space was being rapidly depleted. Although IPv4 addresses began to be rationed out in the early '90s, Network Address Translation (NAT, RFC 1631) and application gateways (AG) have helped to preserve the IPv4 address space, but at some cost to the original goals of IP.

IP was also threatened by youths. The ATM community had this vision of getting rid of the IP layer, and having end-to-end ATM. ATM did not supplant IP, and instead we wound up with a complicated hybrid and two address plans. On top of this, we had more fattening temptations: layer-two tunneling protocols (PPP, LP2T, PPP over Ethernet). "This is progress?" quipped Deering.

Deering talked about lost features of the Internet, properties that were true but are not anymore:

- Transparency – devices that modify addresses (NAT).
- Robustness through fate-sharing – design a system so you do not depend on more resources than you absolutely must. TCP connectiveness depends on state maintained at either end. Add in NAT, and you lose this.
- Dynamic routing. There are many places routing is constrained. With NAT, an organization cannot provide multiple outgoing paths. And the failure of your ISP today means that you cannot simply have a separate path to your network (another ISP).
- Portable addresses. (Early connectors could move their addresses with them, but now addresses are not portable.)
- Unique addresses (NAT).
- Stable addresses (NATs and DHCP).
- Connectionless service. (If you are behind the NAT, NATs do not support transaction-oriented protocols like UDP; protocols do not have an explicit teardown, so NAT boxes just apply their own heuristics.)
- Always-on service, most users of the Internet are behind dialup services, so they must connect or disconnect from the Internet. PPPoE makes cable/DSL like dialup.
- Peer-to-peer communications, not symmetric devices (no servers behind the Net). It becomes the purview of the ISP to set up servers.
- Application independence is another key feature, as more and more application knowledge has to be built into the NAT. If NAT had been widespread before we got the Web, we might have not gotten the Web (no servers behind NAT).

In case all this didn't make the network manager's job hard enough, we renamed bridges to switches. Router people renamed routers as switches (multilayer switches and layerless switches, but this just obscures what is going on). Is this entropy or evolution? Deering believes that this looks like the normal entropy of all large engineered systems over time. "Of course it grows warts and hats over time," said Deering.

What Deering really wants to do is turn hourglass into wineglass – IP over copper, fiber, radio with no intervening protocols. If the lower layers are making IP jobs harder, then let's get rid of lower layers. And we are seeing signs of IP evolving this way. We saw IP over SONET, but people are asking what value does SONET add to IP? We need IPv6 to get back to the narrow shape. Deering concluded, "This is my dream. Only time will tell."

Rob Pike of AT&T Research arose quickly to ask the first question. Pike mentioned that there is not much flow through the stem of the wineglass. He also stated that he thinks that Deering was not giving enough credit to things like what the phone system does for you, and that he doesn't think that IPv6 can replace this. Deering responded by saying that the shape was not supposed to represent a bottleneck, but a reduction in baggage. Pike responded by saying it doesn't appear that IPv6 addresses many of these issues, such as trunking between sites supported by frame relay. Deering responded that circuits could be better based on routing tables, and that MPLS is the current answer to this problem. At this point, Pike suggested that they continue in private.

Jeff Mogul of Compaq Western Research Labs tried to get Deering to talk about firewalls. (It was obvious he doesn't like them.) Eventually, Deering said that firewalls should be in the end host, not in a dedicated firewall (not that we can throw firewalls away today).

Evi Nemeth asked when we will we see IPv6 really deployed. Deering answered "next year," and when Nemeth asked what will get it there, Deering said, "IPv6 in billions of cell phones will require it."

SESSION: OLD DOGS, NEW TRICKS

Summarized by Bob Gray

LEXICAL FILE NAMES IN PLAN 9, OR, GETTING DOT-DOT RIGHT

Rob Pike, Bell Laboratories

Rob Pike pointed out that for 20 years, the UNIX community has been living with the embarrassing problem that chdir("..") doesn't work properly in the presence of symbolic links. From his paper an example is:

```
% echo $HOME
/home/rob
% cd $HOME
% pwd

/n/bopp/v7/rob
% cd /home/rob
% cd /home/ken
% cd ../rob
../rob: bad directory
```

This annoyance, which causes confusion and headaches, should never have lasted so long. The anomaly exists in most versions of UNIX and Plan 9. With a few hours of work, Pike was able to solve the problem for Plan 9 by implementing an "accurate" path name for every active file in the system. It is guaranteed to be the rooted, absolute name the program used to acquire it. A pure lexical definition of chdir("..") is required. That is, take the current working directory and strip off the last component, yielding the new working directory. However, symbolic links introduce an ambiguity in the pathnames. Rob showed that the "correct" name can easily be determined by context.

Rob explained that in Plan 9, "bind" is like symbolic links and a channel is a filesystem handle. All of the necessary information to disambiguate chdir("..") is present in the kernel. A few lines of kernel code implemented the proper solution. Rob challenged the audience to fix this silly bug in contemporary UNIX implementations. He suggested that we may want to look at the Plan 9 source code, which is freely available now at *<http://plan9.bell-labs.com>*.

GECKO: TRACKING A VERY LARGE BILLING SYSTEM

Andrew Hume, AT&T Labs-Research; Scott Daniels, Electronic Data Systems Corp.; Angus MacLellan, AT&T Labs-Research

Andrew Hume described Gecko – an adjunct system to track the efficiency, performance, and accuracy of the AT&T phone billing system. The fundamental question being asked was: "Is every telephone call being billed exactly once?" Their legacy system comprises dozens of big-iron, MVS computers, running hundreds of programs written in Cobol – it was not feasible to interrupt the operations of this 24x7 production system to add tracking software. However, Hume and his team were allowed to tap into the data flows between processes. The sum of these taps amounted to over 250GB/day.

Given the hundreds of millions of transactions per day, Hume's job was to monitor and verify that billing processing was accurate, timely, and complete. His solution was to create "tags" for each telephone call record at each tap point. Nightly, Gecko would add these tags into a giant (60G tag) database, and produce reports highlighting discrepancies from expected flows.

Hume and his team processed the raw information and analyzed the flows using a 32-processor Sun E10000 and two smaller Sun E5000s. They used a number of basic tools: C, ksh, Awk, and Gre (a special implementation of grep). He compared implementations on the Sun and an SGI Origin 2000, and commented that Gecko relied on a solid SMP implementation and they were stymied by inadequacies in PC environments. He found that the SGI gave a 2.5 price/performance advantage over the Sun. But for huge increases in scalability, they would recommend a cluster implementation.

EXTENDED DATA FORMATTING USING SFIO

Glenn S. Fowler, David G. Korn, Kiem-Phong Vo, AT&T Labs-Research

Kiem-Phong Vo described the Sfio package, which is a faster, more robust, and more flexible replacement for the Stdio package. Stdio has several shortcomings in data formatting. For example, to print an abstract scalar such as off_t on some systems you would use a printf specification of %d – on other systems you would need a printf specification of %ld. The Stdio routines, gets and scanf are unsafe when the length of an input line or string exceeds the size of the given buffer. Stdio has no extension mechanism for printing user-defined types. For example, if you had a spatial coordinate type, Coord_t, you would have to use ad hoc formatting methods.

Sfio fixes the above-mentioned problems. It also contains a generalized mechanism for printing arbitrary bases in the range of 2–64. For example, %..2d will print the decimal value of 123 as 1111011.

Here is an example of how Sfio allows flexibility in printing a integer whose size may vary from architecture to architecture:

```
sfprintf(sfstdin, "%I*d", sizeof(intval),
    intval);
```

Similarly, here is a scanning example for floating point numbers:

```
sfscanf(sfstdin, "%I*f". sizeof(fltval),
    &fltval);
```

In both examples, the sizes of the scalar objects determine their types.

Sfio has hooks for handling user-defined types such as complex numbers. This general mechanism allows user to specify format strings, argument lists, and functions to parse each data type.

Sfio outperforms Stdio on all tested platforms mostly due to the new data conversion algorithms. The code is available from *<http://www.research.att.com/sw>*.

INVITED TALK

Implementing 3D Workstation Graphics on PC UNIX Hardware

Daryll Strauss, Precision Insight

Summarized by Matt Grapenthien

3D hardware for PCs has improved to the point where it begins to rival that of traditional 3D graphics workstations. However, providing these capabilities on commodity hardware poses a number of difficult and interesting problems.

After explaining some of the basic algorithms implemented by various 3D hardware, Strauss addressed several of the challenges 3D presents, such as security issues related to commodity hardware. Also, the scheduling granularity used by the Linux kernel, though more efficient for most processes, is too large for smooth video.

Next, Strauss presented the shortcomings of indirect rendering techniques, and presented an alternative: the Direct Rendering Infrastructure (DRI). The DRI is designed to be secure, reliable, and high-performance, by utilizing the available hardware as much as possible. In addition, the DRI is highly modular, allowing different implementations to use only the subset of software they wish.

The DRI is included in XFree86 4.0, and is starting to be used by 3DLabs, HP, and IBM, among others, even on high-end hardware. Precision Insight's work to provide completely open-source solutions to the problems above has shown great promise, even in this somewhat early stage.

FREENIX SESSION: FILE SYSTEMS

Summarized by Kevin E. Fu

Porting the SGI XFS File System to Linux

Jim Mostek, Bill Earl, Steven Levine, Steve Lord, Russell Cattelan, Ken McDonell, Ted Kline, Brian Gaffey, and Rajagopal Ananthanarayanan, SGI

Russell Cattelan talked about the work necessary to port SGI's XFS filesystem to Linux. In particular, he discussed the filesystem interface, the buffer caching, and legal issues. XFS is a highly scalable, journaling filesystem available for free under the GPL.

To maintain atomic updates to the filesystem metadata, XFS keeps a log of its uncommitted actions. Should the machine crash, XFS simply replays the log to recover to a consistent filesystem. Recovery time depends on the size of the uncommitted log. This is in stark contrast to the `fsck` tool, where recovery time depends on the size of the entire filesystem.

A single XFS filesystem can hold as much as 18 million terabytes of data and as much as 9 million terabytes of data per file. XFS is an extent-based filesystem. That is, data are organized into arbitrarily long extents of disk rather than fixed-sized blocks of disk. This allows XFS to achieve a throughput of 7MB/second when reading from a single file on an SGI Origin 2000 system.

XFS uses the vnode/VFS interface. However, Linux does not use this interface. As a result, SGI created the linvfs interface, which maps the Linux VFS interface to the VFS layer in IRIX. At a small overhead cost in translation, this keeps the core XFS code portable.

Second, SGI added pagebuf to Linux. This is a cache and I/O layer to provide most of the advantages from the cache in IRIX. Pagebuf allows pinning of metadata buffers, delayed writes via a daemon, and placement of metadata in a page cache.

Russell also explained the legal process of encumbrance relief. SGI determined, for all the XFS code, which code was original and which code came from third parties. Most of the original XFS code came from SGI. However, XFS contains some software from third parties. A homebrew tool compared every line of code in XFS against the source code from third parties. After removing the third-party code, SGI released XFS under the GPL.

For more information, see *<http://oss.sgi.com/projects/xfs/>*.

Q: Is preallocation fast? A: Yes. You're preallocating an extent of space, not individual blocks. You're not zeroing out files, but you will receive zeros if you read from preallocated space.

Q: How does preallocation work on an API level? A: There are two ways to preallocate, with a command-line utility or new IRCTL calls.

Q: Could you describe what you did at the vnode/VFS layer? A: Our vnode is now part of the inode. The Linux inode would not work with our design. We mostly map functions from linvfs functions to vnode functions.

LinLogFS – A Log-Structured File System for Linux

Christian Czezatke, xS+S; M. Anton Ertl, TU Wien

Christian Czezatke described the LinLogFS filesystem. Started in 1998, LinLogFS aims to achieve faster crash recovery than that of the Ext2 filesystem. LinLogFS also enables the creation of consistent backups while the filesystem remains writable.

LinLogFS evolved from the Ext2 codebase to a log-structured filesystem with better data consistency and cloning/snapshots. It guarantees in-order write consistency.

If you modify part of an Ext2 filesystem during a backup, the change may or may not be noticed. Worse, a simultaneous

backup may only detect parts of the change. LinLogFS does not suffer from this problem because backups can use a snapshot of a filesystem. In this manner, backups are completely atomic while still allowing read-write access to the filesystem.

In the future, Czezatke plans to finish the cleaning program, use less naive data structures and algorithms, and cooperate with the object-based disk project for mirroring.

See
<*http://www.complang.tuwien.ac.at/czezatke/lfs.html*>
for the code.

Q: Have you reviewed the LFS code for the *BSD operating systems? Can you comment on differences or code overlap? A: We looked at BSD LFS and Sprite LFS. We had slightly different goals. We wanted an LFS for Linux. And the Sprite and BSD LFS goal was higher speed than FFS. Our goal was better functionality at the same speed as Ext2.

UNIX File System Extensions in the GNOME Environment

Ettore Perazzoli, Helix Code, Inc.

Ettore Perazzoli spoke about filesystem extensions in the GNOME environment. Perazzoli is an open source software developer and contributor to the GNOME project. GNOME extends the functionality of the UNIX filesystem by using a user-level library called the GNOME Virtual File System.

There are many file formats to juggle: zip, tar, gzip, etc. Each format uses a different tool to view the file contents. GNOME avoids this problem by using a global filesystem namespace. This allows GNOME to identify and view contents of container files such as tar files. Names in the GNOME VFS use an extension of the familiar Web Uniform Resource Identifier scheme.

Ettore discussed how Microsoft Windows has the "My Computer" concept where one can find all system resources such as files, a control panel, printers, and the recycling bin. GNOME VFS addresses the "My Computer" problem by providing filesystem abstractions, and the asynchronous file I/O problem by implementing an asynchronous API. Asynchronous behavior is important because a GUI should be responsive all the time. If the GUI were to block on a file operation, the user cannot stop the operation. GNOME implements asynchronous behavior through an asynchronous virtual filesystem library.

For more information, visit
<*http://www.gnome.org/*> and
<*http://developer.gnome.org/*>.

Q: Why did you choose a pound sign for subschema? A: We inherited this from Midnight Commander. You can always quote the character.

Q: How do you handle symlinks with ".." in the path? A: Symlinks only work within their context. For instance, a symlink inside a tar file will only make sense within the context of the tar file.

Q: Have you considered POSIX AIO as twisted as it may be? A: Yes, but POSIX AIO only lets you make reads/writes from/to the filesystem asynchronous, so it does not help in complex cases such as the .tar or .zip ones, when you have to do other stuff besides physically reading the file.

SESSION: DISTRIBUTION AND SCALABILITY: PROBLEMS AND SOLUTIONS

Summarized by Josh Kelley

Virtual Services: A New Abstraction for Server Consolidation

John Reumann, University of Michigan; Ashish Mehra, IBM T.J. Watson Research Center; Kang G. Shin, University of Michigan; Dilip Kandlur, IBM T.J. Watson Research Center

Virtual services (VSes) provide a way to partition resources on a server cluster between competing applications. VSs can be used to set resource limits and to charge resources out to different virtual services. For example, Web servers for two different companies can be run on a single host by treating them as two separate VSes. Such partitioning has traditionally been done by partitioning a physical host into several virtual hosts. Although this approach provides good insulation between services, it does not allow sharing common subservices between co-hosted sites. VSes address this problem by dynamically adjusting resource bindings. Each request's VS resource context is tracked as the request is handed off across application and machine boundaries, thus allowing VSes to remain insulated from each other while still accessing shared applications and machines.

Implementing VSes involves modifying the operating system's scheduler to partition the CPU. A VS also hooks into various system calls (such as process creation calls and network communication calls) via gates in order to track the propagation of work. Gates adjust resource bindings during system calls (binding a process to a different virtual service as needed) and enforce resource limits. These gates are implemented as loadable kernel modules.

Evaluation shows that virtual services have a minimal negative impact on performance and successfully insulate competing services from each other, even when these services rely on a common shared service. VS support is currently available for Linux 2.0.36; support for Linux 2.2.14 is in development. Source code is available from
<*http://www.eecs.umich.edu/~reumann/vs.html*>.

Location-Aware Scheduling with Minimal Infrastructure

John Heidemann, USC/ISI; Dhaval Shah, Noika

With the increase in use of laptop computers, some way to specify context-dependent configurations and activities could provide a great deal of convenience for users. For example, a user might wish to specify "print to the printer in room 232 when I'm at work" or "disable this background process while I'm on battery." lcron is a context-aware cron that provides a general solution to scheduling such context-dependent activities.

John Heidemann

Context information could come from many sources: GPS receivers, wireless base station name, idle status, network addresses, or battery status are all possibilities. Lcron currently supports the latter two.

The interface to lcron is similar to that of the standard cron and at utilities. A context-dependent `at` command, for example, could be invoked as at 7pm @work. lcron allows users to specify mappings between terms meaningful to the user ('@work') and information readily available to the computer (latitude, longitude, router IP address).

lcron has been in use for two years. It is available from

<http://www.isi.edu/~johnh/SOFTWARE/XCRON>.

Distributed Computing: Moving from CGI to CORBA

James FitzGibbon and Tim Strike, Targetnet.com Inc.

In 1997, TargetNet deployed a banner-ad delivery system using CGI. As the network grew, basic CGI showed itself to be incapable of keeping up with the growth. They wanted a solution that was faster than basic CGI, was freely available, would support distributed computing, and would work with other products. CORBA was chosen as the solution that best met these criteria.

The system that was developed involves two main components. First is an HTTP-to-CORBA proxy. This proxy, called the dispatch server, takes HTTP requests from the Web servers and presents them as CORBA requests to the application servers. This dispatch server allows a three-tier architecture (Web servers, application servers, databases) in which only the Web servers are exposed to the Internet.

The second main component is a method of notifying each dispatch server of the available application servers. A service heartbeat daemon serves this role by maintaining a list of available application servers, providing a client with an available application server when queried, and coordinating with other heartbeat daemons. This design avoids any single point of failure and provides excellent scalability.

CORBA offers an open source, distributed object model, with wide language support. The combination of a dispatch server and a heartbeat daemon allows a three-tier architecture with excellent reliability and near-linear scalability.

INVITED TALK

The Microsoft Antitrust Case: A View from an Expert Witness

Edward Felten, Princeton University

Summarized by Josh Simon

Disclaimer: Neither the author of this write-up nor the speaker is a lawyer.

Dr. Felten was one of the expert witnesses for the U.S. Department of Justice in the recent antitrust case against Microsoft. In his talk he discussed why he believed the government chose him, and he explained the role of an expert witness in antitrust cases.

In October 1997 Felten received an email message from an attorney in the Department of Justice asking to speak with him. After signing a nondisclosure agreement (which is still binding, so he advised us there were some aspects he simply could not answer questions on), and over the course of several months, he spoke with the DOJ until, in January 1998, he signed a contract to be an advisor to the case.

What was the case about? Unlike media portrayals, the case was not about whether Microsoft is good or evil, or whether or not Bill Gates is good or evil, or whether Microsoft's behavior was good or bad. The case was specifically about whether or not Microsoft violated U.S. antitrust laws.

A brief discussion of economics may be helpful here. Competition constrains behavior. You cannot, as a company, hike prices and provide bad products or services when there is competition, for the consumer can go to your competitors and you'll go out of business. Weakly constrained companies, or those companies with little or no competition, have what is called monopoly power. Monopoly power in and of itself is not illegal. What is illegal is using the monopoly power in one market (for example, flour) to weaken competition in another market (for example, sugar).

The U.S. government claimed that (1) Microsoft has monopoly power in the personal-computer operating-system market; and that (2) Microsoft used its monopoly power to (a) force PC manufacturers to shun makers of other (non-Microsoft) applications and operating systems; (b) force AOL and other ISPs to shun Netscape's browsers, Navigator and Communicator; and (c) force customers to install and use Microsoft Internet Explorer. These issues are mostly non-technical and specifically economic. Dr. Felten focused on the technical aspects.

Under U.S. antitrust law, tying one product to another is illegal in some cases. For example, if you have a monopoly on flour, you cannot sell flour and sugar together unless you also sell flour alone. You cannot force customers to buy your sugar in order to get your flour. Similarly, the government argued, Microsoft cannot tie Windows 95 (later, Windows 98) together with Internet Explorer unless it offers both the OS and the browser separately. Microsoft claimed technical efficiencies in bundling the products together.

This boils down to two legal issues. First, what was the motive in combining the OS and the browser? The answer to this is provided by documentation and witnesses, subpoenaed by the government, and not technical. Second, does the combination achieve technical efficiencies beyond that of not combining the two? The answer to this is experimental, technical, based in computer science, and was the focus of the rest of the talk.

Specifically, how did Felten go about testing the efficiencies or lack thereof? He started by hiring two assistants who reverse-engineered both Windows and Internet Explorer. (Note that this work, because it was done on behalf of the government for the specific trial, was not illegal. Doing so yourself in your own basement would be illegal.) After nine months, they were able to assemble a program to remove Internet Explorer from Windows.

The next step in the process was to prepare for court. In general, witnesses have to be very paranoid, nail down the technical details, have sound and valid conclusions, and learn how to be cross-examined. Lawyers, no matter your personal opinion of them, are generally very well-schooled in rhetoric, terminology, and framing of questions, and hiding assumptions in them. They're also good at controlling the topic, pacing the examination, and producing sound bites. In his testimony, Felten demonstrated the "remove IE" program. Jim Alchain, Microsoft vice president, provided 19 benefits of tying the products together and claimed the removal program had bugs. In the government's cross-examination of Alchain, he admitted that all 19 benefits were present if IE was installed on top of Windows 95, and that the video used to show that the demonstration of the removal process had bugs had errors and inconsistencies. Microsoft tried a second demo to show problems with the removal program under controlled circumstances and could not do so. Furthermore, in rebuttal to Microsoft's assertion that the products had to be strongly tied together to gain benefits, the government pointed out that Microsoft Excel and Microsoft Word were not strongly tied and yet were able to interoperate without being inseparable.

Judge Jackson reported in his findings of fact in November 1999 that the combination had no technical efficiencies above installing them separately, Internet Explorer could be removed from the operating system, and tying the browser and the operating system together was, in fact, illegal.

The next phase of the trial was the remedy phase in May 2000. The goals of the remedy phase are to undo the effects of the illegal acts, prevent recurrence of those acts, and be minimally intrusive to the company, if possible. There are generally two ways to accomplish this: structural changes (reorganization or separation of companies) and conduct changes (imposing rules). The judge could choose either or both. The decision was reached to restructure Microsoft such that the operating system (Windows) would be handled by one company and everything else by another. Furthermore, in conduct changes, Microsoft could not place limits on contracts; could not retaliate against companies for competing in other markets (such as, for example, word processing); must allow PC manufacturers to customize the operating systems on the machines they sell; must document their APIs and protocols; and cannot tie the OS and products together without providing a way to remove them.

Microsoft has appealed the case. At the time of this writing it is not clear whether the appeal will be heard in the U.S. Court of Appeals or by the U.S. Supreme Court. The remedies are stayed, or on hold, until the resolution of the appeals or until a settlement of some kind is reached between Microsoft and the U.S. government. Once the case is truly over, Felten's slides will be available on the USENIX Web site.

FREENIX SESSION: SOCKETS

Summarized by Bob Gray

Protocol Independence Using the Sockets API

Craig Metz, University of Virginia

Craig Metz convinced the audience that in an all-IP world, our network programming has become inflexible and uni-protocol. Even though the Berkeley Sockets were designed as a protocol-independent API, other parts of the API (like the name-service functions) grew up as protocol-dependent. So, some of the APIs are not protocol-independent,

which in turn encourages code not to be either. For now, IPv4 is ubiquitous, but as its 32-bit address space runs out, we will find IPv6 more and more compelling. Therefore, it would be prudent to pay attention to the portability issues and even check and retrofit some of our existing network programs.

Metz pointed out multiple problems:

- Hard coded constants in programs,
- Storage limitations and assumptions,
- GUIs that assume four three-digit fields as an address.

For example, many programs hard-code the protocol family as AF_INET, which prevents protocols other than IP from being used. We should not assume a network address will fit in a u_long. And using struct sockaddr_in sin limits programs because it doesn't allocate enough storage for protocols such as IPv6.

Metz recommends using the new POSIX p1003.1g interfaces. For example, getaddrinfo performs the functionality of gethostbyname(3) and getservbyname(3), in a more sophisticated manner.

The new interfaces will take time to be universally deployed; however, they are currently available in at least the following environments: AIX, BSD/OS, FreeBSD, Linux, OpenBSD, NetBSD, Solaris, and Tru64 UNIX. They are expected to be available soon with IRIX and HP-UX.

Scalable Network I/O in Linux

Niels Provos, University of Michigan; Chuck Lever, Sun-Netscape Alliance

Graduate students Niels Provos and Chuck Lever have observed and addressed bottlenecks in Linux where many high-latency, low-bandwidth connections are simultaneously present on a Web-server box. The problem is that the stock kernel data structures and algorithms don't scale well in the presence of thousands of rapidly formed HTTP connections. The file-descriptor selection code needed work. The problem is event notification. It takes a long time for the kernel to find which connections are ready for I/O.

Niels discussed two solutions to remove the inefficiency: the POSIX RT signals API and an optimized poll() along the lines of Banga's declare_interest() interface. He convinced the audience that both mechanisms provide huge improvements for HTTP response time when 251 or 501 idle or inactive connections are present in the background. However, his optimization using /dev/poll yielded the overall best response times.

To achieve high performance, Neils made the following changes to poll():

- Maintain state information in the kernel so that every poll system call doesn't have to retransmit it.
- Allow device drivers to post completion events to poll().
- Eliminate the result copying when poll returns to the user.

The /dev/poll device allows a process to add, modify, and remove "interests" from an interest set. This streamlines poll() calls because only relevant information is passed at system call time. Further, when poll() returns, the application immediately has access to the ready descriptors.

The software enhancements to poll() are freely available.

Accept() Scalability in Linux

Stephen P. Molloy, University of Michigan; Chuck Lever, Sun-Netscape Alliance

Stephen Molloy described the thundering hard problem associated with Linux implementations of the accept() system call. When multiple threads call accept() on the same TCP socket to wait for incoming TCP connections, they are placed into a wait queue. The problem is when a TCP connection is accepted, all of the threads are awakened – even though all but one will immediately need to go back to sleep. As the number of threads goes from a few dozen to hundreds, the kernel begins severe thrashing.

One proposed solution is called Task Exclusive. The idea is to add a flag to the thread state variable, change the handling of wait queues, and connect into a standard, newly added wait-queue mechanism.

Another solution is called Wake One, which adds new calls to complement wake_up() and wake_up_interruptible(). The new functions just wake up one thread when a connection becomes ready.

Molloy presented micro-benchmark data showing huge improvements in "settle time" for both the Task Exclusive and Wake One solutions over the stock kernel.

He also used a Macro-benchmark,- SpecWeb99, to demonstrate the effectiveness of both solutions – over 50% more connections.

The Task Exclusion solution has been incorporated into the Linux kernel. The code is also available at the Linux Scalability Project's home page: <*http://www.citi.umich.edu/projects/linux-scalability/*>.

SESSION: TOOLS

Summarized by Doug Fales

Outwit: UNIX Tool-Based Programming Meets the Windows World

Diomidis D. Spinellis, University of the Aegean

Windows is fundamentally an environment of mouse clicks and pixel output. Very little of this GUI-based OS is accessible to text-based programs. It is with this shortcoming in mind that Diomidis Spinellis developed Outwit. Outwit provides a number of text-based tools for Windows to work together with UNIX-based tools. Specifically, Spinellis' tools provide mechanisms for interaction with

the clipboard, the registry, the ODBC database interface, document properties, and shell links (shortcuts). The presentation included several convincing examples of the time-saving advantages of such an environment.

One example used the winreg command (the interface to the registry) to change the location of the user's home directory from drive C: to drive D:, with a little a help from pipelines and the Win32 version of sed:

```
winreg HKEY_CURRENT_USER |
sed -n 's/C:\\home/D:\\home/gp' |
winreg
```

The winclip tool provides shell-based clipboard access. For example, to copy data from standard input to the Windows clipboard:

```
ls -l | winclip -c
```

and to paste Windows clipboard data to standard output:

```
winclip -p | wc -w
```

Spinellis would be interested in adding functionality to Outwit for new features of Windows 2000, and in providing support for Unicode. The Outwit tools are available at <*http://softlab.icsd.aegean.gr/~dspin/sw/outwit*>.

Plumbing and Other Utilities

Rob Pike, Bell Labs

Plumbing is a Plan9 solution for interprocess communication and message passing between user applications. The general idea is to remove some of the burden on the user of constantly shuffling data from program to program. A common example of this occurs during compilation and debugging, when a compiler generates error messages detailing file location and line number. Plumbing allows the user to access the error in an editor in one click.

Thanks to the pattern-matching language at the core of the plumber's design, it is a far more powerful mechanism than filename-extension associations such as those in Windows. While the goal is for the plumber's default actions to be the most desirable actions, it is highly customizable through a configuration file that defines rules in a pattern-action format.

The actual message passing is relatively trivial because the plumber is a fileserver; messages are written to a file on the server, which then takes appropriate action based on the defined rules. The rules are actually quite flexible and powerful in that they provide a means to interpret the context of messages.

The interface for plumbing is simple and designed to minimize keystrokes and button clicks. The applications themselves do remarkably little work; almost everything is handled within the plumber itself.

One very good application of this sort of automation is in dealing with file formats that might require transformation before viewing, such as an attached Microsoft Word document. With a couple of pattern-matching rules to set up the variables, this rule takes a Word document, converts it to text, and sends it to the editor:

```
plumb start doc2txt $data | \
plumb -i -d edit \
-a action=showdata \
-a filename=$0
```

This one-click approach to conversion and viewing is a slick example of the advantages of this system. Details and more examples may be found in Pike's paper.

Integrating a Command Shell into a Web Browser

Robert C. Miller and Brad A. Myers, Carnegie Mellon University

Rob Miller demonstrated enhancements to his existing Web browser (LAPIS, see <*http://www.usenix.org/events/usenix01/cfp/miller/miller_html/usenix99.html*>). The extension was an integration of browser and command shell. At first glance, this might look like another attempt to unnecessarily GUI-ify a classic typescript tool. However, this project demonstrated some very useful and innovative ways of using a Web browser.

Miller's browser provides a circular redirection of the standard input and output, so that commands are executed as if in one long pipeline. The benefits of such a mechanism in a Web browser are not immediately apparent until the full potential of conventions like the "back" button are realized.

In addition, the browser conveniently separates the standard error and output streams when displaying command output. Some features of the browser are: an embedded pattern-matching language to extract data from a Web page, a command window that can execute traditional shell commands and Web-specific commands, and the ability to automate browsing.

The demonstration showed how LAPIS could be used to visit and print (or save) each page in a document that is strung out over several links. Another interesting application was automating the use of forms. Miller used the pattern-matching language to extract the ISBN of a book from an Amazon.com Web site, and then fed this into a script that had been constructed by LAPIS to consult the form-based Web pages of CMU's library lookup service.

Judging by the demonstration, LAPIS seemed a well-designed, easy to use interface. Furthermore, it does something to alleviate the pain of endless clicking and banner-ad watching that is associated with browsing the Web these days. The browser and its Java source are available at <*http://www.cs.cmu.edu/~rcm/lapis*>.

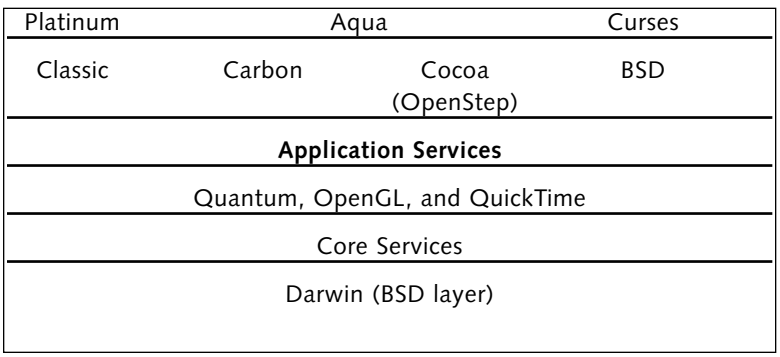
INVITED TALK

Challenges in Integrating the Mac OS and BSD Environments

Wilfredo Sanchez, Apple Computer

Summarized by Josh Simon

Fred Sanchez discussed some of the challenges in integrating the MacOS and BSD environments to produce MacOS X. Historically, the Mac was designed to provide an excellent user interface ("the best possible user experience") with tight hardware integration and a single user. In contrast UNIX was designed to solve engineering problems, using open source (for differing values of "open"), running on shared multi-user computers and with administrative overhead. There are positives and negatives with both approaches. MacOS X is based on the Mach 3.0 kernel and attempts to take the best from both worlds. A picture may help explain how all this hangs together:



Sanchez next talked about four problem areas in the integration: filesystems, files, multiple users, and backwards compatibility. Case sensitivity was not much of an issue; conflicts are rare and most substitutions are trivial. MacOS uses colon as the path separator; UNIX uses the slash. Path names change depending on whether you talk through the Carbon and Classic interfaces (colon, :) or the Cocoa and BSD interfaces (slash, /). Filename translation is also required, since it is possible for a slash to be present in a MacOS file name. File IDs are a persistent file handle that follows a file in MacOS, providing for robust alias management. However, this is not implemented in filesystems other than HFS+, so the Carbon interface provides for nonpersistent file IDs. Hard links are not supported in HFS+, but it fakes it, providing the equivalent behavior to the UFS hard link. Complex files – specifically, the MacOS data and resource forks – are in the Mac filesystems (HFS+, UFS, and NFS v4) but not the UNIX filesystems (UFS and NFS v3). The possible solutions to this problem include using AppleDouble MIME encoding, which would be good for commands like cp and tar but bad for commands using mmap(), or using two distinct files, which makes renaming and creating files tough, overloads the name space, and confuses cp and tar. The solution they chose was to hide both the data and resource forks underneath the filename (for example, filename/data and filename/resource, looking like a directory entry but not a directory) and have the open() system call return the data fork only. This lets editors and most commands (except archiving commands, like cp and tar, and mv across filesystem boundaries) have the expected behavior. Another filesystem problem is permissions (which exist in HFS+ and MacOS X but not in MacOS 9's HFS). The solution here is to base default permissions on the directory modes.

The second problem area is files. Special characters were allowed in MacOS filenames (including space, backslash, and the forward slash). Filename translation works around most of these problems, though users have to understand that "I/O stuff" in MacOS is the same as "I:O_stuff" in UNIX on the same machine. Also, to help reduce problems in directory permissions and handling they chose to follow NeXT's approach and treat a directory as a bundle, reducing the need for complex files and simplifying software installations, allowing drag-and-drop to install new software.

The third problem area involves multiple users. MacOS historically thought of itself as having only a single user and focused on ease of use. This lets the Mac user perform operations like setting the clock, reading any file, installing software, moving stuff around, and so on. Currently MacOS X provides hooks for UID management (such as integrating with a NetInfo or NIS or LDAP environment) and tracks known (UNIX-like) and unknown disks, disabling commands like chown and chgrp on unknown disks.

The fourth and final problem area Sanchez discussed was compatibility with legacy software and hardware. Legacy software has to "just work," and the API and toolkit cannot change, so previous binaries must continue to work unchanged. The Classic interface provides this compatibility mode. Classic is effectively a MacOS X application that runs MacOS 9 in a sandbox. This causes some disk-access problems, depending on the level (application, filesystem, disk driver, or SCSI driver). The closed architecture of the hardware is very abstracted, which helps move up the stack from low-level to the high-level application without breaking anything.

Questions focused on security, the desire to have a root account, and the terminal window or shell. The X11 windowing system can be run on MacOS X, though Apple will not be providing it. Software ports are available from <*http://www.stepwise.com/*>. Additional details can be found at <*http://www.mit.edu/people/wsanchez/papers/USENIX_2000/*>, <*http://www.apple.com/macosx/*>, and <*http://www.apple.com/darwin/*>.

FREENIX SESSION: NETWORK PUBLISHING

Summarized by Matt Grapenthien

The growth and evolution of the Internet has caused some of its limitations to become more acute. The presenters in this session attempt to address some of these problems with more dynamic and generally more useful systems and protocols than those currently in wide use.

PERMANENT WEB PUBLISHING

David S. H. Rosenthal, Sun Microsystems Laboratories; Vicky Reich, Stanford University Libraries

David Rosenthal presented LOCKSS (Lots of Copies Keep Stuff Safe), a system to preserve access to scientific journals in electronic form. Unlike normal systems, LOCKSS has far more replicas than necessary just to survive the anticipated failures. Exploiting the surplus of replicas, LOCKSS allows much looser coordination among them.

THE GLOBE DISTRIBUTION NETWORK

A. Bakker, E. Amade, and G. Ballintijn, Vrije Universiteit Amsterdam; I. Kuz, Delft University of Technology; P. Verkaik, I. van der Wijk, M. van Steen, and A. S. Tanenbaum, Vrije Universiteit Amsterdam

The Globe Distribution System, presented by Arno Bakker, attempts to address the problem of distribution to a worldwide audience through selective, per-document replication, as opposed to the "all-or-none" replication policies currently in use. Though still in an early stage, the project's goal of a "better" WWW/FTP seems promising.

OPEN INFORMATION POOLS

Johan Pouwelse, Delft University of Technology

Johan Pouwelse presented a method to allow modification and extension of existing Web pages by allowing public write access to collections of WWW-based databases. Open Information Pools further address the problem of quickly evaluating huge amounts of content, through an open rating and moderation system. Experiments on similar, already-existing systems seem to prove these concepts very valuable.

SESSION: KERNEL STRUCTURES

Summarized by Josh Kelley

OPERATING SYSTEM SUPPORT FOR MULTI-USER, REMOTE, GRAPHICAL INTERACTION

Alexander Ya-li Wong and Margo I. Seltzer, Harvard University

An increasing number of services are being provided over the network instead of locally. Examples include filesystems (NFS), storage (Fibre Channel), memory (Distributed Shared Memory), and interfaces (thin clients). Of these, thin clients are often neglected.

The key characteristics of thin-client service are that it is interactive, multi-user, graphical, and remote. One of the most important features of thin-client service is low latency. This presentation compared two operating systems, Windows NT 4.0 Terminal Server Edition, and Linux 2.0.36 running the X Window System, and examined how well they provide these characteristics.

The first two questions regarding thin-client service are processor management and memory management. The OS's goal should be to prevent the user from experiencing any perceptible latency. To minimize latency, the OS should insure that interactive performance is background-load-independent and should swap out pages belonging to interactive processes last. Although NT offers special support for scheduling interactive threads, experiments showed Linux to be much better, both for processor scheduling and for low latencies while swapping pages in from disk. Open questions here include the best time slice to use for interactive processes and how the Linux kernel can identify interactive threads (since interactivity is determined in user space) for special treatment.

A third question regarding thin-client service is network load. In experiments, NT's Remote Display Protocol presented a much lower network load, with a larger message size, than did the X or LBX protocols used by Linux and X Windows. (This may be due partially to poorly coded X applications.) RDP's use of a client-side bitmap cache allowed it to have virtually no load in the specific area of animation (as long as the cache was not overloaded). These experiments point out the importance of a client-side cache.

TECHNIQUES FOR THE DESIGN OF JAVA OPERATING SYSTEMS

Godmar Back, Patrick Tullmann, Leigh Stoller, Wilson C. Hsieh, and Jay Lepreau, University of Utah

A Java OS is an execution environment for Java bytecode that provides standard operating-system functionality: separation and protection, resource management, and interapplication communication. It may run on a traditional OS or it may be embedded in an application. The purpose of a Java OS is to support executing multiple Java applications.

There are several options for arranging the operating system, Java Virtual Machine, and Java applications. One approach is physical separation: one OS per one JVM per one app. Such an approach is expensive, prevents embedding of an OS within an application, and makes communication difficult. A second approach is separate JVM processes running under one OS. This approach has inefficient resource use, requires a underlying OS, and makes for difficult communication and no embedding within outside applications. A third approach is an ad hoc layer that supports running multiple applications within one JVM. Examples of this approach include an applet context or a servlet

engine. However, this approach offers insufficient separation between the processes, with no resource control and unsafe termination of one runaway process. The fourth, and best, approach is to provide support for processes within the JVM, turning the JVM into a Java OS. The Java OS can be designed to replace the base OS as well.

There are several design decisions to be made for a Java OS. One example is the area of shared memory management. Issues here include the precision of accounting, the ability to reclaim shared objects, and the need for full reclamation of memory upon process termination. Java's automatic garbage collection complicates these issues. Solutions to the question of shared memory management include copying (simple but slow), indirect sharing via revocable proxies, direct sharing via a dedicated shared heap, and a hybrid approach of both direct and indirect. The J-Kernel, from Cornell University, and Alta and K0 (which later became KaffeOS), both from the University of Utah, all offer different solutions to this question and illustrate the general tradeoffs between separation, resource management, and communication.

Signaled Receiver Processing

José Brustoloni, Eran Gabber, Abraham Silberschatz, and Amit Singh, Lucent Technologies-Bell Laboratories

This session presented signaled receiver processing, an alternative to the BSD's traditional IP packet receiver processing. Since implementations of and derivatives of BSD have appeared on a variety of platforms, BSD receiver processing is a part of many operating systems, although it has several disadvantages. Protocol processing of received packets in BSD is interrupt-driven. This results in scheduling anomalies; CPU time spent processing packets is charged to the currently running process or is not charged at all. Therefore, no quality of service (QoS) guarantees are possible. A second problem is receive livelock, in which the system spends all of its time processing incoming packets, even when no buffer space is available to store these packets.

One alternative to BSD receiver processing is lazy receiver processing (LRP). To prevent receive livelock, LRP uses earlier demultiplexing to detect full receive buffers as soon as possible and drop packets accordingly. UDP packets are processed synchronously, at the receive call. TCP packets are processed asynchronously, via an extra kernel thread per process or via a system-wide process that uses resource containers. (Resource containers are an abstraction used to separate resource principal and protection domain. Resources used by kernel-level code can be charged out to user processes.) This processing of UDP and TCP packets allows LRP to avoid the BSD receiver processing's scheduling anomalies.

There are several disadvantages with LRP. First, it does not work on systems that do not implement kernel threads or resource containers. Second, under LRP, TCP is always asynchronous and shares resources equally with the application. Third, LRP is designed for hosts, not gateways. Its early demultiplexing is too simplistic for gateways, and time-sharing scheduling is inadequate for gateways. There are open questions with LRP regarding how well it would work with realtime schedulers and proportional-share schedulers.

Signaled receiver processing (SRP) is presented as an alternative method that avoids these problems with LRP. When a packet arrives, the OS signals the receiving application. By default, the packet is processed asynchronously, although the application may instead choose to catch, block, or ignore the packet, in order to defer processing until the next receive call. SRP processes incoming packets in several stages; only the actual hardware input is handled at the interrupt level. Processing is handled via a multi-stage early demultiplexer, or MED, and is transferred from one stage to another via the next stage submit (NSS) function. The NSS function signals the application by sending it a SIGUIQ (unprocessed interrupt queue).

Performance tests show that throughput, CPU utilization, and round trip times are practically the same for SRP under Eclipse/BSD as they are for BSD receiver processing under FreeBSD. Tests also show that SRP successfully prevents receive livelock. It is easily portable, allows for flexible scheduling and use with gateways, and still allows for QoS guarantees.

INVITED TALK

The Convergence of Networking and Storage: Will it be SAN or NAS?

Rod Van Meter, Network Alchemy

Summarized by Josh Simon

The goal of this talk was to provide models for thinking about SANs and NASs. Network-attached storage (NAS) is like NFS on the LAN; storage area networks (SAN) are like a bunch of Fibre Channel–attached disks.

There are several patterns of data sharing, such as one-to-many users, one-to-many locations, time slices, and fault tolerance; activities, such as read only, read-write, multiple simultaneous reads, and multiple simultaneous writes; and multiple ranges, of machines, CPUs, LAN versus WAN, and known versus unknown clients.

When sharing data over the network, how should you think about it? There are 19 principles that Levy and Silverschatz came up with that describe the file. These include the naming scheme, component unit, user mobility, availability, scalability, networking, performance, and security. There is Garth

Gibson's taxonomy of four cases: server-attached disks, like a Solaris machine; server integrated disks, like a Network Appliance machine; netSCSI, or SCSI disks shared across many hosts with one "trusted" host to do the writes, and network-attached secure devices (NASD). Over time, devices are evolving to become more and more network-attached, smarter, and programmable.

Van Meter went into several areas in more detail. Access models can be application-specific (like databases or HTTP), file-by-file (like most Unix file systems), logical blocks (like SCSI or IDE disks), or object-based (like NASD). Connections can be over any sort of transport, including Ethernet, HiPPI, Fibre Channel, ATM, SCSI, and more. Each connection model is at the physical and link layers and assumes there is a transport layer (such as TCP/IP), though other transport protocols are possible (like ST or XTP or UMTP). The issues of concurrency (are locks mandatory or advisory, is management centralized or distributed?), security (authorization and authentication, data integrity, privacy, and nonrepudiation), and network ("it doesn't matter" versus "it's all that matters") all need to be considered.

Given all those issues, there are three major classes of solutions today. The first is a distributed file system (DFS), also known as NAS. This model is a lot of computers and lots of data; examples include NFS v2, AFS, Sprite, CIFS, and XFS. The bottleneck with these systems is the file manager or object store; drawbacks include the nonprogrammability of these devices and the fact that they are OS-specific and have redundant functionality (performing the same steps different times in different layers).

The second class of solution is storage area networks (SAN). These tend to have few computers and lots of data and tend to be performance-critical. These are usually contained in a single server or machine room; the machines tend to have separate data and control networks. These devices' drawbacks are that they are neither programmable nor smart, they're too new to work well, they provide poor support for heterogeneity, and the scalability is questionable. However, there is a very low error rate and the application layer can perform data recovery. Examples of SANs include VAX clusters, NT clusters, CXFS from SGI, GFS, and SANergy.

The third solution class is NASD, developed at CMU. The devices themselves are more intelligent and perform their own file management. Clients have an NFS-like access model; disk drives enforce (but do not define) security policies. The problems with NASD is that it's too new to have reliable details, more invention is necessary, there are some OS dependencies, and some added functionality may be duplicated in different layers. Which solution is right for you? That depends on your organization's needs and priorities.

FREENIX SESSION: X11 AND USER INTERFACES

Summarized by Gustavo Vegas

The GNOME Canvas: A Generic Engine for Structured Graphics

Federico Mena-Quintero, Helix Code, Inc.; Raph Levien, Code Art Studio

The GNOME Canvas is a generic high-level engine for structured graphics. A canvas is a window in which things can be drawn. It contains a collection of graphical items such as lines, polygons, ellipses, smooth curves, and text. Graphics on this canvas are deemed to be structured because you can place these geometric shapes in the canvas and later on access the objects to change their attributes, such as position, color, and size. The canvas is in charge of all redrawing operations.

The GNOME canvas has an open interface that permits applications that use the canvas to create their own custom item types. Thus, the canvas can work as a generic display engine for all kinds of applications. The GNOME canvas items are GTK+ objects derived from an abstract class (GnomeCanvasItem), that gives the methods for objects to be implemented. Using the GTK+ object system provides several advantages, such as the possibility of associating arbitrary data items to canvas items.

The GNOME canvas also uses the Libart library for its external imaging model in antialias mode. Libart is a library that provides a superset of the PostScript imaging model, and it provides support for antialiasing and alpha transparency. The end result is that graphics' contours are smoothed out to eliminate jagged edges.

Several applications that are currently distributed as part of the GNOME environment use the GNOME canvas to render graphics and other types of data. Examples of such applications are Gnumeric (the GNOME spreadsheet), GNOME-PIM (personal information manager), and Evolution (the next-generation mail and groupware program for GNOME).

For more information about the GNOME project and the GNOME canvas, see <*http://developer.gnome.org/*>.

Efficiently Scheduling X Clients

Keith Packard, SuSE Inc.

Keith Packard presented a new scheduling algorithm for X11. The technical motivation behind this project is that the original scheduling mechanism in X11 is simplistic and can potentially starve interactive applications while a graphics-intensive program runs. This program is evident when one runs programs like plaid, which generate many rendering requests that can tie up the system for long periods of time, making it unusable

by other programs, as simple as an xterm.

The X server is typically a single-threaded network server that uses well-known ports to receive connections from clients for which it processes requests sent over the port. To detect pending input from clients, it uses the select(2) system call. When the set of clients with pending input has been determined, the X server starts executing the requests, starting with the smallest file descriptor. Each client uses a buffer to read some of the data from the network connection. This buffer can be resized but it is typically 4KB. The requests are executed until either the buffer is exhausted of complete requests, or after ten requests. After this has taken place, the server figures out if there are any clients with pending complete requests. If this is the case, the server stops the select(2) call and goes back to process those pending requests. When all client input is exhausted, the X server calls select(2) again to await for more data. The problem with this algorithm is that it gives preference to more active clients. If a client generates complex requests, these requests may take up more time to be satisfied and this client will end up hogging the server. As a consequence, clients that generate fewer requests are starved in the presence of more active clients. Also, clients that do not generate a complete request during their turn will be ignored. On the positive side, when clients are busy, the server spends most of its time executing requests and wastes little time on system calls.

The design goal of the proposed solution is to provide relatively fine-grained time-based scheduling with increased priority given to applications that receive user input. Each client would be given an initial priority at connect time. Requests are executed from the client with the highest priority, and various events may change the priority of a given client. This system intends to penalize overactive clients and praise clients with little activity.

In the performance measurements that were presented a measurable change was apparent. However, the two schedulers were within only 2% of each other. This shows that the changes in the scheduler have little impact on the tool used for performance measurement. The tool used was X11perf, a widely available tool. This tool runs with very little competition from other clients, and thus most of the benefits of the new scheduler go unnoticed.

In conclusion, simple changes on the scheduler, based on real-life observations of the X server behavior, can bring some advantages over the original scheduler without impacting performance negatively. For more information and for work that has been incorporated into the 4.0 release of the X Window System from the XFree86 group, see <*http://www.xfree86.org/*>.

The AT&T AST Open Source Software Collection

Glenn S. Fowler, David G. Korn, Stephen S. North, and Kiem-Phong Vo, AT&T Laboratories-Research

David Korn presented a suite of tools that have been released to the open source community by AT&T. This tool suite includes widely known components that may or may not be directly used in graphics applications, but the fact that these have been released has a profound impact in the open source community.

These tools have been released under an AT&T license agreement. This is not GPL or LGPL, so it is important to read the license if one is going to make use of any of the software components for any purpose.

The components of this suite are deemed to be highly portable to practically any environment, given the right base. They may not necessarily be complete applications, but they can be reusable tools to produce other powerful pieces of software. Their focus when creating this software suite is reusability. They have also focused in creating software libraries that encompass core computing functions such as I/O and memory allocation and other new algorithms and data structures such as data compression and differencing and graph drawing. Thus, they created libraries like,

Libast – Porting base library for their software tools.
Sfio – This I/O library provides a robust interface and implements new buffering and data formatting algorithms that are more efficient than those in the standard I/O library, Stdio.
Vmalloc – This memory-allocation library allows creation of different memory regions based on application-defined memory types (heap, shared, memory mapped, etc.) and some library-provided memory-management strategies.
Cdt – This container data-type library provides a comprehensive set of containers under a unified interface: ordered/unordered sets/multisets, lists, stacks, and queues.
Libexpr – This library provides run-time evaluation for simple C-styled expressions.
Libgraph – This graph library supports attributed graphs, generalized nested subgraphs, and stream file I/O in a flexible graph data language. It is built on top of the `Cdt` library and employs disciplines for I/O, memory management, graph-object name-space management and object-update callbacks. This library is the base of the Graphviz package.

Other complete tools that have been released as part of this collection are reimplementations of programs like the KornShell language and nmake, and other new applications like: `tw`, a more powerful find and xargs; and warp, a tool that helped with Y2K testing by running

a process through it and simulation a time and clock speed different from the actual system's time and clock.

For more information, please see <*http://www.research.att.com/sw/tools/*>. For the AT&T source code license agreement, please see <http://www.research.att.com/sw/license/ast-open.html>.

SESSION: WORKS IN PROGRESS

Summarized by Kevin Fu

Perl on the Itanium

Murray Nesbitt <murray@activestate.com>, ActiveState Tool Corp.

Nesbitt discussed his experiences in building Perl under Windows 2000, Linux, and Monterey on the 64-bit Intel Itanium. It was straightforward to build Perl on IA-64 Linux. Building Perl on Monterey was almost as easy; it required a standard Perl hints file to compile though. Murray's main point was that building Perl on Windows 2000 was more painful for a number of reasons such as lack of configure-script support and problems with type sizes and abstractions. In the future Murray plans to work on optimization. In short, it's fairly easy to port Perl, but it's helpful to share an office with a Perl guru.

ttf2pt1: A TTF to Adobe Type 1 Font Converter

Sergey Babkin <babkin@users.sourceforge.net>, Santa Cruz Operation

Babkin described his work on a TTF-to-Adobe Type 1 converter. The converter also attempts to clean the outlines from detects and automatically generate Type 1 hints. His program optimizes the method to make the conversion look good. It comes under the BSD license and is reasonably well modularized. He noted that this work has nothing to do with SCO and is simply his personal hobby. For more information, see <*http://ttf2pt1.sourceforge.net/*>.

lodd: A Pipelining and In-Pipe Data Manipulation Tool

Joseph Pingenot <jap3003@ksu.edu>, Kansas State University

Pingenot, an undergraduate at KSU, talked about a tool to combine multiple channels of input and output. The lodd utility hopes to perform tasks such as acting as a logical dd and mixing three pipes together. lodd 1.x has stream-pipe support and can mix multiple pipelines together, manipulate data at the block level, perform bitwise logic, and divide pipelines. lodd is dd-compatible. Some of the interesting issues in creating lodd include dealing with blocking I/O, backstreams, block sizes, shell limitations, and deciding what to do if a pipe closes. Visit <*http://www.phys.ksu.edu/~trelane/lodd*> for more information.

Q: Are functions extensible? A: Not in the preliminary version. Hopefully in the future.

Q: How does one use lodd within a shell? A: I am now looking at ways to implement a shell syntax. Suggestions are welcome.

vstack: Easily Catch Some Buffer Overrun Attacks

Craig Metz <cmetz@inner.net>, University of Virginia

Buffer overruns typically work by overwriting a function's return address with a value of the adversary's choice. In this way, an adversary can change the flow of control to execute, for example, a root shell. Metz described a simple approach that prevents many commonly exploited overruns.

vstack verifies that function return addresses do not change. It does so by keeping a separate virtual stack of return addresses and frame pointers. On return from a function, a program verifies that the return address and frame pointer on the execution stack matches that on the virtual stack. If not, the program jumps to a fault handler. This does not break standard calling convention and requires changes only to the caller convention in a compiler.

A sample Perl implementation exists for the x86. It edits assembly code to insert the checks and management of the virtual stack. The code will appear soon under a BSD-style license. The performance loss is minimal. In the future, Metz plans to have more sophisticated choices in what to do after detecting an overrun. vstack does not catch every overrun attack, but it can catch the vast majority.

Q: Instead of verifying the return address matches, why not simply use the return address on the virtual stack? A: There might be other corrupted data. In special cases it might be OK to use the virtual stack directly, but not in general.

Q: What prevents overwriting the virtual stack itself? A: It is elsewhere in memory. If you can overwrite an extent of 2^{32} space, then you can overwrite everything anyway.

Q: Does longjmp() or other functions that unwind the real stack confuse vstack? A: Probably. (Offline Metz explained that this is mostly solvable by using wrappers around longjmp() and related functions. It won't catch every case, but it will catch most of them.)

Q: Can I overwrite data on the stack? A: Yes, but vstack will detect changes made to return addresses.

kq: Kernel Queues in FreeBSD

John-Mark Gurney <jmg@freebsd.org>, FreeBSD

Kernel queues are a stateful method of event notification. Instead of passing which events to monitor each time as is done with select(2) and poll(2), the program tells the kernel which events need notification. kq supports event monitors (filters) for file descriptors, processes, signals, asynchronous I/O, and VNODEs. State is allocated in kernel memory. kq

pays attention to what file descriptors are ready for reading and writing.

John-Mark described l0pht's watch program modified to use kq. The watch program looks for temporary files created in /tmp. Before using kq, the watch program consumed a lot of CPU time by polling directory entries in /tmp over and over. With kq, you get a notification when the /tmp directory changes. In this manner, the watch program does not needlessly scan an unmodified directory.

Efforts are underway to use kq in the Squid HTTP proxy cache for asynchronous I/O and in ircd to reduce the server load. Visit <*http://people.freebsd.org/~jmg/kq.html*> for more information.

Q: How does this compare to /dev/poll on Solaris 8? A: I haven't looked at /dev/poll; it's hard to say. However, my system is extremely lightweight.

Q: If a given source sends multiple signals, will it cause a series of kernel memory allocations before the read occurs? A: The memory is actually allocated when you register.

Q: Can you unify with other kernel name spaces? A: Currently you are limited to a filesystem. However, pretty much any kernel object can be associated.

Autonomous Service Composition on the Web

Laurence Melloul <melloul@stanford.edu>, Stanford University

The goal of this service it to allow dynamic specification of composition requests. The advantages include a cost-effective development cycle, better fault tolerance, and high availability. Melloul chose the Web as the medium because it has autonomous services, uses a public infrastructure, is common, is simple, and speaks a language independent protocol (HTTP). The two main issues are detection of service interface changes and verification of semantic compatibility between service parameters. The key is to build on the ontology by web users.

The Hummingbird File System

Liddy Shriver <shriver@research.bell-labs.com>, Bell Labs, Lucent Technologies

The Hummingbird File System caters to workloads of a caching Web proxy. On UNIX machines, server software such as Apache or Squid typically runs on a derivative of the 4.2BSD Fast File System. FFS was not designed with the workload of a proxy server in mind. In particular a Web-proxy workload has high temporal locality, relaxed persistence, and a read/write ratio different from most workloads. FFS also includes features which a proxy server does not require.

The Hummingbird File System takes advantage of Web-proxy server properties such as whole file access, small files on average, and repeatable reference locality sets. It also co-locates the storage of an HTML Web page with its embedded GIFs. Performance measurements show that under a sample Web-proxy workload, Hummingbird supports throughput five to ten times greater than that of XFS and EFS (SGI) and six to ten times greater than that of FFS mounted asynchronously (FreeBSD).

Future work includes persistence of data. At the moment, Hummingbird does not worry about persistence because the data can be regenerated from origin servers. Visit <http://www.bell-labs.com/project/hummingbird/> for more information.

Q: Does Hummingbird support for HTTP reads for specific byte ranges? A: Not yet.

Q: When does Hummingbird write to disk? A: During idle time and when memory is filled and space is needed.

The Self-Certifying File System

Kevin Fu <fubob@mit.edu>, MIT Lab for Computer Science

The Self-Certifying File System is a secure, global filesystem with completely decentralized control. SFS lets you access your files from anywhere and share them with anyone, anywhere. Anyone can set up an SFS server, and any user can access any server from any client. SFS lets you share files across administrative realms without involving administrators or certificate authorities.

SFS is a secure network filesystem in the sense that it provides confidentiality and integrity of the Remote Procedure Calls (RPCs) going over the wire. SFS runs at user level and uses NFSv3 for portability. It performs between TCP and UDP NFS on FreeBSD and is in day-to-day use for Fu's research group.

Server public keys are made explicit in pathnames. The pathname to a remote file includes a "HostID" that consists of a cryptographic hash of a server's public key and hostname. At a high level, the HostID is essentially equivalent to a public key. Using this convention, we can easily implement certificate authorities with symbolic links. Then a certificate authority can chain trust with symbolic links. A system of user agents and authenticated lists of trusted HostId takes care of most of the HostIds.

There is also a read-only dialect suitable for highly replicated, public, read-only data (e.g., software-distribution or certificate authorities). In this scheme, an administrator creates offline a signed database of a filesystem to export. Untrusted servers can replicate this database. Clients can then select any of the untrusted servers. Because the database is signed and the self-certifying path denotes the corresponding public key, the client can verify that the data is authentic. Measurements show that a read-only server can handle many times the workload of a read-write SFS server and that server-side authentication is

more than an order of magnitude faster than that of SSL.

SFS is free software. The SFS developers have used it on OpenBSD, FreeBSD, Solaris, OSF/1, and a patched Linux kernel. Download the software from <*http://www.fs.net/*>.

Q: Why not store server public keys in a file? Is the self-certification just a hack? A: We believe the symlink approach is more elegant. And c'mon! It's a cool hack.

Q: Have you thought about committing this to the FreeBSD tree? You should make sure that your software stays maintained by contacting someone in charge of distributions for each operating system. A: You are certainly welcome to include SFS in your operating system. We can try to locate the appropriate contacts for each operating system, but you are welcome to contact the SFS developers too. Email <*sfs-dev@pdos.lcs.mit.edu*>.

Q: Do you rely on DNS for security? A: No. We only use DNS as a hint to locate a server. If a fake SFS server responds to a request, the client will detect the fake because the fake server's private key will not correspond to the public key described in the self-certifying path. You could receive notification of such failures via an agent. In our system, the worst an adversary can do is deny service.

NFS Version 4 Open Systems Project

Andy Adamson <andros@umich.edu>, CITI, University of Michigan

Adamson discussed some of the interesting features of NFS version 4. The CITI group at the University of Michigan received funding from Sun Microsystems to implement NFSv4 on Linux and OpenBSD.

NFS version 4 has compound RPCs that perform multiple operations per RPC. There is no more mountd. There is no more lockd. Locking is incorporated into the protocol and includes DOS share locks and nonblocking byte-range locks with lease-based recovery. Delegation aids in client file cache consistency. The server controls who gets delegation. Security is added to the RPC layer via GSSAPI. NFSv4 requires Kerberos5 and Lipkey PKI implementations. The security mechanism and QOP is negotiated between the client and server.

The code has passed all nine basic Connectathon tests under Linux. The CITI folks have just started work on the OpenBSD code. In the near future, Adamson plans to rebase to Linux 2.2.4 and finish the OpenBSD port. Source code will be available by September 1, 2000. Visit <*http://www.citi.umich.edu/projects/nfsv4/*> for more information.

Q: Is there backwards compatibility with NFS3? A: There is none.

Q: Do you expect future growth in NFS specifications? A: Ohhhh yeah.

Q: What are the terms of the license? A: Under Linux it is GPLed. Under OpenBSD it has the OpenBSD license.

Q: Does it work under IPv6? A: We'd love it to work with IPv6. Would you like to financially support us?

Alfa-1: A Simulated Computer with Educational Purpose

Alejandro Troccoli <atroccol@dc.uba.ar> and Sergio Zlotnik <szlotnik@dc.uba.ar>, University of Buenos Aires

The Alfa-1 project consists of software tools to simulate a processor. This helps in teaching computer architecture to undergraduate students. The GAD tool was used as a basis for developing a simulated computer, allowing students to experiment with the approach defined by the DEVS formalism. The model is based mainly on the specification of the Integer Unit of the Sparc processor.

Undergraduates implemented most of the system, which is now completely specified and implemented. In the future, the Alfa-1 staff hopes to add a GUI, perform exhaustive testing, use digital logic gates, add another level of cache memory, and promote educational use of Alfa-1. Visit <*http://www.dc.uba.ar/people/proyinv/usenix/*> for more information.

Q: What are you able to simulate? A: If we had enough computational power, we could simulate everything.

Q: Are there triggers, tracepoints, GDB for this? A: This is plain C code. You can debug the simulator using any standard tool.

Tellme Studio

Jeff Kellem <composer@tellme.com>, Tellme Networks

Tellme Studio offers a free service for developing and testing voice XML applications. All you need is knowledge of VoiceXML and JavaScript (if you choose to use JavaScript). You write your VoiceXML code, put it up on a Web server somewhere, log in to Tellme Studio (<*http://studio.tellme.com/*>), and give the URL pointing to your code. You are then given an 800 number to call to immediately test out the application. Tellme Studio includes VoiceXML documentation, code and grammar examples, and a community for sharing ideas.

Passwords Found on a Wireless Network

Dug Song <dugsong@monkey.org>, CITI, UMich

Receiving a standing ovation and giving by far the most entertaining talk at the conference, Dug Song from the CITI group at the University of Michigan gave a "brief report of what he found in the air." He further described tools he created to demonstrate the insecurity of his network. In the process, the audience convinced him to give a live demonstration on how easy it is to collect passwords and shadow a user's Web surfing.

Song's second slide included slightly sanitized sniffer logs. Among the finds were cleartext root logins via Telnet and colorful passwords such as "hello dug song, do I smell." While explaining an ebay.com URL, Dug reasoned that, "I don't know if Matt Blaze is here, but it sure looks like he is."

On to more serious stuff, Dug explained the rationale behind his seemingly malicious behavior. He views himself not as a bad guy, but as someone promoting "security through public humiliation." On that note, he introduced the mother of all password sniffers and a few penetration testing tools. His penetration tools include:

arpredirect – This tool is extremely effective for sniffing on switched Ethernet. It does so by poisoning ARP. It politely restores the ARP mappings when finished.
macof – This is a C port of a tool to flood a network with random MAC addresses. It causes some switches to fail open in repeating mode. This effectively turns the switch into a hub for the purposes of sniffing. Dug commented, "Switch becomes hub, sniffing is good."
tcpkill – A evil tool to selectively kill connections. However, Dug uses it to remotely initialize connection state.
tcpnice – This selectively slows down traffic by using ICMP quenches and shrinking TCP window sizes. This is useful for sniffers that work better on slower network traffic. It's also useful against things like Napster.
dsniff – This sniffer decodes 30 major protocols from Telnet to Meeting Maker. The HTTP module recognizes password URL schemes for many e-commerce sites (e.g., Web mail sites, eBay, etc). dsniff uses a magic(5)-style automatic protocol detection. For example, running Telnet on port 3000 will not fool dsniff because it can determine protocol by analyzing the traffic content.
filesnarf – Sucks down cleartext NFS2 and NFS3 traffic. Very useful against files such as .XAuthorithy and .ssh/identity. This is Song's "motivation" for developing NFSv4.
mailsnar – A fast and easy way to violate the Electronic Communications Privacy Act of 1986 (18 USC 2701-2711), be careful. This snarfs cleartext mail and outputs the contents into a convenient format suitable for offline browsing with your favorite mail reader.
urlsnarf – Same idea as mailsnarf but for URLs.
webspy – Very sinister. It allows you to watch someone's Web surfing in real-time. Dug demonstrated the tool by shadowing the Web surfing of an audience member.

Song concluded that many people incorrectly believe wireless and switched networks are immune to sniffing. He thinks that public humiliation can remind people of this misconception. Visit *<http://www.monkey.org/~dugsong/dsniff>* for more information. The slides are on *<http://www.monkey.org/~dugsong/talks/usenix00.ps>*.

Q: Should we block access to port 23 at USENIX terminal rooms? Then cleartext Telnet sessions will not happen. A: That's a technical solution to a social problem. I like my way better.

Q: The conference network ran out of DHCP leases. Can your tools help me? A: During the conference I wrote a short "dhcpfree" program to forcibly free up IP addresses. [Crowd laughs as he shows the code.]

Q: Once upon a time in a terminal room, I measured the ratio of SSH/Telnet traffic. At one point, I said loudly "look at all the interesting passwords!" All of a sudden, the ratio went up. A: Yup.

INVITED TALK

Lessons Learned about Open Source

Jim Gettys, Compaq

Summarized by Matt Grapenthien

In this interesting, informative, and thoroughly entertaining talk, Jim Gettys addressed the history of several projects and presented the most (and least) successful methodologies over a time frame of about two decades.

Beginning with a history of X, Gettys traced its peaks and valleys from "prehistory" (1983) through our present "baroque" period. When the CDE (or "cruddy desktop environment") took over the market, X development became almost nonexistent. Then, starting in

Jim Gettys

about 1996, a combination of factors revitalized X. X is better today than any point in the past, and the future looks promising.

Next Gettys talked about several individual projects, tracing the Apache's market share through the past decade. Through these case studies, he noted which practices worked best (release continuously, make it easy for developers to contribute) and which didn't work at all.

Gettys achieved a rare balance of keeping the talk both very entertaining and very useful. His wit, sarcasm, and experience (20 years with OSS) made this a valuable and enjoyable session.

SESSION: RUN-TIME TOOLS AND TRICKS

Summarized by Josh Kelley

DITools: Application-level Support for Dynamic Extension and Flexible Composition

Albert Serra, Nacho Navarro, and Toni Cortes, Universitat Politècnica de Catalunya

DITools is a way to modify an application or library without access to the source code. It can be used, for example, to profile a binary without instrumentation, to use another library with a program without rewriting the program, or to fix a bug in a library without recompiling it.

A process image is traditionally built of three parts: a runtime loader, a main program, and one or more libraries. The OS brings all of the needed modules into the address space, and then the runtime loader resolves references between these modules. DITools augments the loader to insert an extension backend between the main program and the libraries. The main program then calls the extension backend instead of the libraries, and the backend forwards calls to the libraries as appropriate.

DITools loads before entering the program. It offers dynamic loading support and uses binding management to interpose modules. DITools can change function bindings on a per-module basis, through rebinding, or globally, through redefinition. DITools also offers two interposition modes; it can change the linkage table to point directly to a backend's wrapper, or it can change the linkage table to point to DITools's dispatcher, which calls callback functions in the backend before and after calling the original function. To ensure correct operation, DITools transparently checks for and handles events that may affect its behavior, such as dynamic loading, process forking, and multithreading.

Performance tests show that DITools incurs a reasonable overhead on function calls. DITools complements related methods such as static binary rewriting or dynamic instrumentation based on code patching. DITools operates at a higher abstraction level and is simpler than these related methods, but it works at the function level only. It presents an application-level tool to intercept cross-module references and easily make changes. DITools is available from <*http://www.ac.upc.es/recerca/CAP/DITools*>.

Portable Multithreading-The Signal Stack Trick for User-Space Thread Creation

Ralf S. Engelschall, Technische Universität München (TUM)

Multithreading offers many advantages to a programmer, but finding a portable fallback approach to implement threading on UNIX platforms can be difficult, if the standardized Pthreads API is not available. Setjmp(3) and longjmp(3) are the traditional methods of transferring execution control in user-space, but they do not address the question of how to create a machine context on a particular runtime stack. The ucontext(3) API allows for user-space context creation and switching, but it is still not available across all platforms.

A solution to this problem is to use the UNIX signal-handling facilities in conjunction with setjmp(3) and longjmp(3). The process can create a machine context by setting up a signal stack using sigaltstack(2), then sending itself a signal to transfer control onto that stack. Once in that stack, the process saves the machine context there via setjmp(3), leaves the signal handler scope, and later restores the saved machine context without signal-handler scope. Then it finally enters the thread-startup routine while running on this particular stack. A much more detailed description of the algorithm is available in the author's paper or at <*http://www.engelschall.com/pw/usenix/2000/*>).

Performance tests show that thread creation using this signal stack trick is about 15 times slower than thread creation using ucontext(3), because of the signaling required. However, user-space context switching is as fast as with ucontext(3). The signal-stack trick offers an extremely portable method of implementing user-space threads without relying on assembly code or platform-specific facilities. This fallback approach is used in the GNU Portable Threads (Pth) library, available from <*http://www.gnu.org/software/pth/*>.

Transparent Run-Time Defense against Stack-Smashing Attacks

Arash Baratloo and Navjot Singh, Bell Labs Research, Lucent Technologies; Timothy Tsai, Reliable Software Technologies

Buffer overflows are one of the most common sources of security vulnerabilities. Crackers can exploit buffer overflows to achieve two dependent goals: injecting attack code and altering control flow to execute this attack flow. The basic method of exploiting a buffer overflow is the stack-smashing attack, where the attacker puts the attack code on the stack, then overwrites the current function's return address with the address of the attack code. This presentation offered two complementary defenses against stack-smashing attacks.

The first defense is libsafe, which intercepts calls to unsafe functions and replaces them with safe versions. The majority of buffer overflows result from the misuse of unsafe functions such as strcpy and fscanf. At runtime, libsafe estimates a safe upper bound on the stack buffer. It intercepts calls to these unsafe functions and replaces them with calls using this upper bound, thus containing overflows to a safe region and guaranteeing that the stack return addresses are protected. The function-interception technique is similar to that used by zlib.

The second defense, libverify, uses binary rewrites to ensure that return addresses are valid before use. It wraps each function to save the function's return address on the heap upon entry and check the return address at exit. If the return address has changed, then the process displays an error to screen and syslog and dies. libverify uses runtime instrumentation (copies the program at runtime and changes it) to insert its stack-checking code. This technique is similar to that used by StackGuard, but without recompilation of the source code.

Both libraries can be loaded for an already-compiled binary using an entry in /etc/ld.so.preload or the LD_PRELOAD environment variable. Testing showed that these libraries successfully prevented known exploits on several programs with reasonable execution time overhead. libsafe is available for Linux from <*http://www.bell-labs.com/org/11356/libsafe.html*>.

INVITED TALK

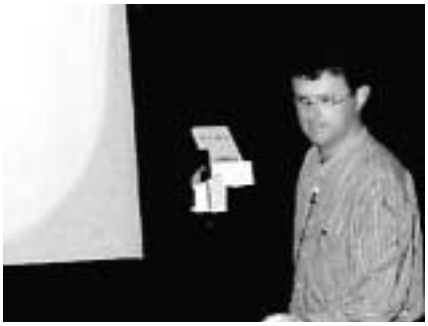
An Introduction to Quantum Computation and Communication

Rob Pike, Lucent Technologies – Bell Labs

Summarized by Doug Fales

Rob Pike's discussion of quantum computing was a very forward-looking, change-of-pace invited talk. He first reviewed some quantum mechanics to bring the audience to common ground. The always-popular polarized-light experiment helped to demonstrate the principles. He also presented the famous two-slit experiment, in which a single photon passed through two very small slits in a barrier still creates interference on the opposite side of the barrier. Pike used this as a demonstration of the Quantum Measurement Postulate because when the particle is measured to see which slit it passed through, the pattern disappears.

After the simplified (but challenging) introduction, Pike progressed into the more specific field of quantum computation. He addressed quantum-mechanical phenomena like decoherence and entanglement, as well as the implementation of quantum computational systems (qubits, quantum gates, etc.). The most promising aspects of this infant technology, those of massive parallelism and



Rob Pike

zero-energy calculation, were clarified. As examples of the power of quantum computation, Pike went over the possibilities of Shor's algorithm for factoring large primes in polynomial time, and Grover's algorithm, which searches a list in square root time.

In addition to the computational side of quantum mechanics, Pike also addressed possibilities for communications, which he saw as not so distant in the future. This included a discussion of EPR pairs (a pair of entangled photons produced by electron-positron annihilation, named after Einstein, Podolsky and Rosen) and how entanglement is actually an advantage for communications.

Perhaps the most interesting point in the whole talk was the theme that information is known to be a physical quantity, restricted by a law of conservation, much like energy or mass. Furthermore, as Moore's Law continues to shrink classical computers, we will run into a physical barrier of size; as Pike put it, "we're running out of particles."

Although quantum computation is still relatively far from real application, Pike noted in closing that we cannot tell yet whether progress in this field will be linear or exponential. After all, he said, classical computers were as much of an enigma 60 years ago as quantum computation is today. The slides for this presentation are at <*http://www.usenix.org/events/usenix2000/invitedtalks/pike_html/index.html*>.

INVITED TALK

Providing Future Web Services

Andy Poggio, Sun Labs

Summarized by Josh Simon

Andy Poggio basically expanded on Bill Joy's keynote talk. The Internet has effectively begun to mimic Main Street and is beginning to provide those services that Main Street cannot, such as any time and anywhere. The six Webs are of relevance:

Near web – Monitor, keyboard, and mouse attached to a nearby system; personalized news such as multimedia and news-on-demand; and educational aspects like multimedia, interactive simulations, and so on. An example of educational uses of the near web can be found at <*http://www.planetit.com/techcenters/docs/*>.

Far web – The television or appliance with remote control, providing entertainment on demand; multiple data sources, providing a lower barrier to entry; possibly targeted advertising with product placement in on-demand movies showing Coke ads on the sides of a taxi cab to Coke drinkers but showing Pepsi drinkers a Pepsi ad in the same position.

Voice web – For use when the hand and eye are busy, like driving a car.

e-commerce web – Computer-to-computer, such as auctions (both "forward" like eBay and "reverse" like eWanted) and dynamic pricing.

Device web – Device-to-device for non-PC devices like cell phones and pagers and set-top boxes. These include agents to collect data and remote distributed processing via IP and Java.

Here web – Personal digital assets, like "my CDs" or "my MP3s" or "my DVDs"; providing on-demand access and ownership, and allowing the end-user to create her own environments.

So how do we get there? Three aspects need to be worked on. First, the network has to be enhanced. IPv6 provides more address space, better configuration management, authentication, and authorization, but adoption has been slow. Poggio predicts wired devices will win over wireless devices, both quality of service and overprovisioning will continue, optical fiber will replace or supercede electrical (copper) wiring, and the last mile to the home or the consumer will be fiber instead of ADSL or cable modems or satellites. Second, the computer-chip architecture will probably remain based on silicon for the next ten or so years. Quantum effects (see the "Quantum Computing" talk for more information) show up around 0.02 microns, so we need new approaches such as optical computing, organic computing, quantum computing, or computational fogs (virtual realities). Third, Poggio believes that the system architecture will connect three components – CPU server, storage devices, and the network – with some form of fast pipe, probably InfiniBand (a high-bandwidth, low-error, low-latency fast interconnect).

FREENIX SESSION: COOL STUFF

Summarized by Jeff Schouten

An Operating System in Java for the Lego Mindstorms RCX Microcontroller

Pekka Nikander, Helsinki University of Technology

The RCX Microcontroller is a device sold as part of a Lego set. It's designed to move a bit of Lego here or there, and making a mostly functional robot for a child to play with. As a project, a Java operating system was developed for this microcontroller by Pekka Nikander and his students at Helsinki University of Technology. The RCX consists of a Hitachi H8 microcontroller, 32K of ram, an LCD panel, an IR transceiver, and several IO devices.

Using mostly Java, a small bit of C++, and a bit of H8 assembly, there is a mostly functional OS for the RCX. When the RCX agrees to take the download (about one in three times) it runs fairly well, but gets stuck in a loop once in a while. A student is currently debugging this behavior. (That gave us a few laughs).

In all, it's an amazingly small OS, for a very limited task, but it works – it can be done.

LAP: A Little Language for OS Emulation

Donn M. Seeley, Berkeley Software Design, Inc.

LAP (Linux Application Platform), is a Linux-emulation package for BSD/OS that allows Linux applications to run under BSD/OS. By loading a shared library on a BSD system to "catch" Linux system-level calls and reroute them to the BSD kernel, a BSD user can run a Linux application.

Emulation is quite successful. A number of interesting Linux applications run via LAP and liblinux under BSD, including Adobe Acrobat Reader v4, Netscape Communicator, and WordPerfect 8. Not all functions of a Linux system are emulated, most notably the Linux clone() system call.

Overall, it seems to not eat a lot of processor, and it provides BSD users a bit of flexibility they didn't previously have.

Traffic Data Repository at the WIDE Project

Kenjiro Cho, Sony CSL; Koushirou Mitsuya, Keio University; Akira Kato, University of Tokyo

The idea behind this project is to collect statistical data on trans-Pacific backbone links on the Internet. WIDE is a Japanese research consortium that designed this data repository, with the intent of building free tools with which to build your own repository.

One problem with this type of data is privacy, another security. How do they protect private information from leaking into the repository and thus generating a possible security breach? Removal of the payload is the first step, leaving only header information to analyze. Address Scrambling is the second step, rewriting or stripping out IP addresses out of ICMP and TCP packets.

INVITED TALK

The GNOME Project

Miguel de Icaza

Summarized by Josh Kelley

Although Linux has proved itself on the server, its progress on the desktop lagged until quite recently. Three years ago, UNIX had little innovation; the last significant user interface change was X Windows. There was little code reuse and no consistent way to build desktop applications. Improvements were incremental enhancements to speed and feature lists rather than major architectural changes, and there was little direction between groups making these enhancements. The

GNOME project aims to correct these shortcomings.

The GNOME project is a unified, concerted effort to build a free desktop environment for UNIX. One major part of this effort is GNOME's component platform. GNOME is designed as a collection of components that build on top of one another; dependencies between components are encouraged, and each application or component exports its internals to others via CORBA.

This approach to writing software as a collection of small components works well with free software. Since free-software contributors tend to come and go, a set of components allows them to focus on, and contribute to, a small problem with relatively little ramp-up time.

Traditional approaches to components have several disadvantages. UNIX command-line tools may not be easy to use and can only communicate through unidirectional pipes that transfer primarily streams of textual data. Object-oriented programming has its advantages, but sharing objects among different object-oriented languages is difficult.

Bonobo is the GNOME solution for components. Bonobo is a component architecture based on CORBA and partially inspired by Microsoft's COM/ActiveX/OLE2. Bonobo provides the building blocks and the core infrastructure for writing and using components. Bonobo can be divided into two parts: the CORBA interfaces that are the contract between the providers and the users, and the implementations of those interfaces. Other implementations of these interfaces are possible. For example, KDE could choose to provide these interfaces to allow KDE and GNOME components to work together.

Since Bonobo is based on CORBA, it has all of the standard CORBA features, including language independence and support for automation and scripting. Bonobo's basic interface is Bonobo::Unknown, which provides two basic features: life-cycle management of an object through reference counting, and dynamic discovery of features through a standard query interface. Specific components may support any number of additional interfaces to allow full access to their functionality.

One application of Bonobo would be to provide standard interfaces to system services. Traditional UNIX, for example, stores configuration information in a variety of formats, mostly in files under the /etc directory, and specific instructions on how to make changes and how to put these changes into effect often differ. The goal of Bonobo (and GNOME) is to have CORBA interfaces everywhere, for every service, for the desktop, and for each application. The entire system should be scriptable. Applications should have easy access to one another's functionality rather than having to write desired functionality themselves. This provides better IPC (a more flexible alternative to pipe and fork) and better Internet protocols (applications can communicate via Bonobo rather than using ad hoc protocols).

Bonobo is used in several applications that are either in development or are currently available. These include Gnumeric (a spreadsheet), Sodipodi (a draw application), Evolution (a mail and calendar system), and Nautilus (the GNOME 2.0 file manager). Bonobo is also integrated with the rest of GNOME; the GNOME canvas, for example, allows embedding of objects of any type, and the GNOME printing architecture offers functionality similar to the canvas to provide an alternative to using PostScript for everything. GNOME 1.4, which includes Bonobo 1.0, should be released this October.

FREENIX SESSION: SHORT TOPICS

Summarized by Craig Soules

JEmacs – The Java/Scheme-Based Emacs

Per Bothner

The first talk of the "shorts" section of Freenix was a discussion of JEmacs, the Java/Scheme-based Emacs. Described as "A next-generation Emacs based on Java," JEmacs offers all of the standard features of Emacs, as well as a number of useful new features.

The rationale behind JEmacs is that Java implementations have become increasingly faster, making it suitable to an application such as Emacs. Additionally, it offers a number of useful features, such as built-in Unicode support and multithreading. Through the use of Kawa, JEmacs also has an easy-to-use Scheme interpreter that offers features of Java, such as Swing (its GUI interface), in a simple scripting language.

In order to make the transition to JEmacs as painless as possible, JEmacs also offers full ELisp support. Although there are some slight difficulties with integrating ELisp with the multithreaded nature of Java, the author seems to have them well in hand. For more information on JEmacs see <*http://www.jemacs.net/*>. Kawa (JEmacs's Scheme implementation) also has a home page, <*http://www.gnu.org/software/kawa/*>.

A New Rendering Model for X

Keith Packard, SuSE, Inc.

Keith Packard spoke on developing a new rendering model for the X Window System. The current X system was developed based upon the PostScript specification of the time, and was targeted at being a simple solution to support simple "business" applications. Today most programs don't even use many of the available features of X, relying on inefficient or unaccelerated external libraries

to handle screen drawing, which is done mostly client side. By looking at the requirements of most X programs today, Packard has developed a new model for X that will offer many of its missing features.

Although many things in the current X system are lacking, there are a few things that are worth keeping, such as testable pixelization and exposed pixel values. It is important also, that all of the potential, good and bad, of the old model is left in place for use by legacy programs. The proposed solution offered here is simply to add a number of new features. These include: alpha composition, antialiasing support, a finer-grained coordinate system, more rendering primitives, and better text handling.

UBC: An Efficient Unified I/O and Memory Caching Subsystem for NetBSD

Chuck Silvers, The NetBSD Project

UBC is a unification of the I/O and memory caching systems of NetBSD, and was presented by Chuck Silvers. Unlike many other current operating systems, NetBSD had a separated page and file caching. This led to many complications within the kernel involving management between the two systems to avoid stale data and proper behavior. Additionally, the page cache has a number of useful features not available in the file-buffer cache, such as being dynamically resized.

The proposed solution to this problem is to have the page cache manage all file access. This offers a number of benefits: only one copy of the data will ever be cached, which also prevents any copying to avoid stale data; the page cache can dynamically resize itself; and cached data no longer needs to be constantly mapped in memory to remain in the cache. This is all managed through several new calls, which are used by the buffer cache to retrieve and manage pages from the virtual-memory system.

Although this new system has the potential to have increased performance, as well as reducing overall cache size, many improvements need to be made in order to make this a reality. Tests with the initial system indicate worse sequential-access performance than the current caching system. This is claimed to be due mostly to unaggressive read ahead and bad pager algorithms. In addition to the performance tuning, several other enhancements are in the future, such as soft-updates support, avoiding the need to map pages in order to do file I/O, and page loan out. The current implementation of this code will become available in the release following the 1.5 release of NetBSD.

Mbuf Issues in 4.4BSD IPv6 Support – Experiences from the KAME IPv6/IPsec Implementation

Jun-ichiro itojun Hagino, Internet Initiative Japan, Inc.

This presentation looked at the difficulties found in offering IPv6 support in a BSD environment. Although IPv6 was designed with the idea that it can easily be layered over current IPv4 implementations, its implementation on 4.4BSD proved to be more complicated than advertised.

These complications arose from a number of IPv4-specific assumptions made within the BSD kernel. The biggest of these problems arose from a change in header size in IPv6, which led to severe packet loss. In order to fix this problem, the author created a new parser for the IP layer which not only offered correct IPv6 support, but also reduced the amount of copying and `mbuf` allocation significantly. This work has now been integrated with all of the major BSD kernels available. For more information on this work, see *<http://www.kame.net/>*.

malloc() Performance in a Multithreaded Linux Environment

Chuck Lever and David Boreham, Sun-Netscape Alliance

This work, presented by Chuck Lever, is a direct result of the Linux scalability project at the University of Michigan. This project aims to enumerate the problems facing network servers and ensure that Linux offers support greater than or equal to current production-level operating systems for network servers.

Because malloc() is a major concern for network servers, it is important that it be able to offer acceptable performance in situations that require multithreaded asynchronous I/O, low latency and high data throughput with a predictable response time, and a potentially unbounded input set of unpredictable requests. It has been found that the manner in which malloc() performs can have a significant effect in overall system performance. One example showed a 6x performance degradation when the iPlanet LDAP server ran on four-way hardware with a native implementation of malloc().

To study the performance of malloc(), three benchmarks were created. The first compared the elapsed runtime of N threads accessing the same heap concurrently. It was found that Linux outperformed Solaris for N greater than two. The second benchmark, which tested unbounded memory consumption, allocated an object in one thread and free the same object in the second thread. It was found that Linux has no pathological heap growth in this situation. The final benchmark tested data placement within the heap by allocating a data object normally with malloc() and then letting several threads write into it many times. Bad data placement causes poor performance, especially on SMP hardware. Linux's version of malloc() aligns data to eight-byte boundaries, which resulted in widely varying application

performance on CITI's four-way test machine. In conclusion, the version of malloc() offered with glibc 2.1 showed overall acceptable performance on two- and four-way hardware, but still needs work and special attention to reduce performance and scalability problems caused by sloppy data placement.

CLOSING SESSION

New Horizons for Music on the Internet

Thomas Dolby Robertson

Summarized by Josh Simon

Thomas Dolby is a musician (you'll probably remember him from "She Blinded Me with Science!") who's been working for at least 20 years on integrating computers into music. (One historical tidbit: The drums in "Science!" were actually generated by a discotheque's light-control board.)

Dolby is one of the founders of Beatnik (<*http://www.beatnik.com/*>), a tool suite or platform to transfer descriptions of the music, not the music itself, over the Internet. For example, the description would define which voice and attributes to use, and the local client side would be able to translate that into music or effects. This effectively allows a Web page to be scored for sound as well as for sight.

For example, several companies have theme music for their logos that you may have heard on TV or radio ads. These companies can now, when you visit their Web sites, play the jingle theme without needing to download hundreds of kilobytes, merely tens of bytes. Similarly, a Web designer can now add sound effects to her site, such that scrolling over a button not only lights the button but plays a sound effect. Another use for the technology is to mix your own music with your favorite artists, turning on and off tracks (such as drums, guitars, and vocals) as you see fit, allowing for personalized albums at a fraction of the disk space. (In the example provided during the talk, a 20K text file would replace a 5MB MP3 file.) In addition to the "way cool" and "marketing" approaches, there's an additional educational component to Beatnik. For example, you can set up musical regions on a page and allow the user to experiment with mixing different instruments to generate different types of sounds.

The technical information: Beatnik combines the best of the MIDI format's efficiency and the WAV format's fidelity. Using "a proprietary key thingy" for encryption, Beatnik is interactive and cross-platform, providing an easy way to author music. And because the client is free, anyone can play the results. The audio engine is a 64-voice general MIDI synthesizer and mixer, with downloadable samples, audio file streaming, and a 64:2 channel digital mixer. It uses less than 0.5% of a CPU per voice, and there are 75 callable Java methods at runtime. It supports all the common formats (midi, mp3, wav, aiff, au, and snd), as well as a proprietary rich music format (rmf), which is both compressed and encrypted with the copyright. RMF files can be created with the Beatnik Editor. (Version 2 is free while in beta but may be for-pay software in production.) The editor allows for access to a sound bank, sequencer, envelope settings, filters, oscillations, reverbs, batch conversions (for example, entire libraries), converting loops and samples to MP3, and encryption of your sound. And there is an archive of licensable music so you can pay the royalties and get the license burned into your sample.

Web authoring is easy with the EZ Sonifier tool, which generates JavaScript; middling with tools like NetObjects' Fusion, Adobe GoLive, and Macromedia Dreamweaver; and hard if you write it yourself, though there is a JavaScript-authoring API available for the music object.

Beatnik is partnered with Skywalker Sound, the sound-effects division of Lucasfilms Ltd.

BOF SESSION

Workplace Issues for Lesbian, Gay, Bisexual, Transgendered and Friends

Summarized by Chris Josephes and Tom Limoncelli

Although the first submitted name for this BoF suggested that it was for "sysadmins," it was well-attended by managers, engineers, programmers, and anyone else who felt like attending. The LGBT BoF has had a long history at USENIX and LISA conferences. The crowd was an even mix of newcomers and conference veterans. The purpose of the session was to give people to opportunity to talk about their work environments, and to provide the opportunity to network with other attendees whom they may not have otherwise met during the conference.

Everyone introduced themselves and the companies they work for. Then they talked about how their employers have handled issues facing LGBT employees and related experiences they may have had when dealing with their employers. As it is becoming harder to find qualified people, more companies are adapting their benefits to the LGBT community in hopes of attracting new talent and retaining existing employees.

Of the 34 attendees, most reported that their employers established a nondiscrimination policy that included sexual orientation. On top of that, some employers also offered domestic-partnership benefits for registered partners. Another issue employers are trying to address is maintaining a safe, friendly, nonthreatening environment for employees by implementing peering programs, diversity training, and employee groups.

Andrew Hume, USENIX vice president, attended the BoF to welcome everyone

and assure us that USENIX is committed to creating a space that is inviting and supportive to all attendees. His comments were well-received.

After the introductions were finished, the BoF became an open floor. We talked more in depth about the issues of employment and some of the recruiting plans some firms were offering. Several attendees, all from one large company, reported that they now have a recruitment effort targeting the LGBT community since they feel their accepting work environment is one of their competitive advantages. Someone pointed out which BoF attendees were presenting papers at the conference, so that others could attend and lend support. Ideas for improving attendance at future BoFs were brought up, including a couple of suggestions for making sure the male/female ratio was more representative of the conference attendance.

When it was time for the BoF to officially end, everyone agreed to informally get together at one of the hospitality suites scheduled for that night.

Trip Report

USENIX ANNUAL TECHNICAL CONFERENCE

by Peter H. Salus

<peter@matrix.net>

I had a great time at the 25th Anniversary USENIX conference in San Diego. Oh, boy! San Diego! Right. I didn't even get off the hotel premises for nearly three days.

After all, there were Dennis Ritchie, Ken Thompson, Bill Joy, Rob Pike, Kirk McKusick, Eric Allman, Tom Christiansen, Elizabeth Zwicky, Margo Seltzer, Sam Leffler, Jim Gettys, Clem Cole, Evi Nemeth, Mike Ubell, Teus Hagen, Rich Miller, Oz Yigit, Miguel de Icaza, and over 2000 others inside the hotel.

In fact, I only got to about two papers/presentations a day. Take Thursday, the "middle" day of the conference. From 9:00 to 11:00 am, I listened to Ed Felten of Princeton University talk about the Microsoft trial. As he was under DoJ secrecy rules (he was adviser to and expert witness for the DoJ), there was much he couldn't say. What he could say was fascinating. Then I walked around the exhibits, and chatted. Then I went to a publications-committee meeting where I was a guest of the USENIX board. Then I was on the Dr. Dobb's Webcast for over an hour, to be succeeded by Linus Torvalds. So I chatted with Linus, his wife, and their two little girls. Then I sat down with some folks from Sleepycat to learn about embedded databases. And it's now after 5:00 pm. At 6:00, I went to the celebratory reception; at 8:00, I went to hear Linus at the Linux BoF. At nine, I went to the "Old Farts' BoF." At 10:30 pm I met my wife, who wanted to go out for dinner. I was too tired. We stayed in the Marriott.

On Wednesday morning I had the pleasure of witnessing the awarding of the "Flame" (for lifetime achievement) to the late Rich Stevens, with Rich's wife, children, and sister getting a five-minute standing ovation from the attendees. Then Bill Joy shared his thoughts about the future of computing.

Bill traced his beginnings in the field – just about 25 years ago at Berkeley – and traced the origins of "open source" to the bases of university research and of UNIX on the PDP-11. While there was the question of whether doing software is "research," the common answer was "yes," and as researchers publish results, there could be no property rights adhering to that research. Lately, largely because of commercial and industrial contributions, there are more and more "entanglements."

Bill noted that he had written "a really boring Java book." My guess is that he was referring to *The Java Language Specification* by Joy, Steele, Gosling, and Bracha. As I really liked the first edition (1996) and found the second edition (2000) even better – when was the last time the second edition of a reference book was smaller than the first? – I think I'd disagree.

One of Joy's more interesting comments turned on the fact that UNIX was inherently reliable because of its modularization. The consequence is that "Microsoft is clearly foolish" in attempting to make all of its applications integral and the construct monolithic. "Microsoft is beyond retrograde," Bill said. "All interfaces should be published."

Turning to the future, Bill scorned the notion of the death of Moore's law. In fact, he thinks that we might see another "ten-to-the-sixth" improvement over the next 30 years, just as machinery has improved a million times over the past 30. Bill thinks that molecular computing will enable us to come to grips with the "grand challenge problems" like those of cell biology. Some of these steps will come about through algorithmic improvements, some through greater emphasis on remote storage and fast transmission.

However, he pointed out, the more that's done remotely, the higher the toll for the round-trip, even with high-speed optical connections. "The Internet isn't about packets," he said. "It's about end-to-end."

On the client-server side, Bill said that he saw six Webs in the future:

the "near web," which we currently use;
the "far web," the entertainment for couch potatoes;
the "here web," of the pocket device and the cell phone;
the "weird web," involving smart clothing and voice-activated cars;
and two "invisible" webs: "e-business" and "truly pervasive."

There was lots more. It was a fine hour.

Then I spent over an hour on the Dr. Dobb's Webcast.

Wednesday night there was a four-hour BSD BoF (that's right) organized by Kirk McKusick. An hour each of OpenBSD, FreeBSD, NetBSD, and BSDI. I sat through over an hour of it. While there were pockets of enthusiasts cheering and jeering, it was a generally highly intelligent, well-informed group of about 400. And I thought it exciting to see them all together. A doff of my cap to you, Kirk.

Thursday morning, as I mentioned, I went to hear Ed Felten. While most folks know many of the details of the case, I found listening to the recollections of a participant truly fascinating. My personal feeling is that the case is more about economics and business than about technology, but there are (clearly) two technical queries of significance: (1) is there a technical advantage to tying Windows to the browser? and (2) can they be disentangled without injury to either?

The clear answer to (1) is "no" (there is, of course, a business advantage for Microsoft) and Felten himself demonstrated the answer to (2) in court. In fact, advocates of the small kernel (like Linus Torvalds) as well as "old-timers" (like Bill Joy) all shun the interwoven monolithic monstrosities produced by Microsoft.

Went off for another hour of Webcast.

The "Old Farts' BoF" on Thursday night was very well attended. Ken Thompson, Lou Katz, Greg Rose, Dennis Moomaw, Clem Cole, and I were among the recollectors. Toward 10:00 pm, Professor Arun K. Sharma, head of computer science and engineering at the University of New South Wales, announced a "drive" to raise $A2M (= $US1.2M) to establish a John Lions Chair of Operating Systems at UNSW. As I considered John a teacher and a friend, I handed Arun my check for $1,000 on the spot. I found it very moving.

Friday morning I was spellbound as Rob Pike, who is always exciting to hear, spoke about quantum computing. This is the kind of paper that differentiates a real conference from a hawker's sideshow: Heisenberg, Schroedinger, integral signs, real equations. Rob said that we should rid ourselves of the notion that "the elements of information are independent" and get used to concepts like "conservation of information." "A quantum computer is probabilistic," he remarked. "It's not going to happen soon," but it will happen.

Now we can turn it over to Gibson, Sterling, Stephenson, and Vinge . . .

A fine conference. I can't wait till the 30th Anniversary.

The 25th Anniversary Reception was a good party!

java performance

Proxy Classes

by Glen McCluskey

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

Proxy classes are a new Java feature, one that allows you to create dynamic classes at runtime. These classes can be used to add a level of indirection to interface method calls, so that calls can be trapped and processed as desired. Using this feature, it's possible to do call tracing for performance purposes, add functionality to existing interfaces, and so on.

We'll start by reviewing some basics of Java interface programming.

Interface Programming

In Java programming, an interface is an abstract type, typically a set of methods that specifies a contract for the type. For example, a List interface might look like this:

```
interface List {
    void add(Object o);
    void del(int i);
    Object get(int i);
    int size();
}
```

In this example, methods are specified for adding, deleting, and retrieving elements from a list, and for obtaining the list size.

An interface itself contains no implementation. That is, you do not say:

```
interface List {
    int size() {...}
}
```

Rather, you implement an interface via a Java class, like this:

```
class LinkedList implements List {
    int size() {...}
}
```

The interface itself does not specify any details of how a list is to be represented (such as with a vector or linked list); this decision is left to an implementing class. Thus the interface specifies a contract, and the class implements the contract.

You can program in terms of interface types, for example:

```
List lst = new LinkedList();
```

and then say things like:

```
int sz = lst.size();
```

without worrying about exactly which class (ArrayList, LinkedList, etc.) is used to implement the List interface, or the details of list representation.

Proxies

java.lang.reflect.Proxy is a class that allows you to create proxy classes at runtime, classes that implement specified interfaces. Once you have an instance of a proxy class, you can use it to provide a level of indirection when you make interface method calls. Calls to methods of interfaces implemented by the proxy class are dispatched to a single invoke() method in an invocation handler. At this point, you can intercept the calls, for example to provide method logging for performance analysis purposes.

In other words, you create a proxy class and instance by specifying a set of interfaces, and an invocation handler to call when any of the methods in the interfaces are invoked. You then use this instance as a proxy for some other object; the proxy sits on top of the other object and allows you to trap method calls that you would normally make on the underlying object.

When a method call is trapped, you are supplied with the signature of the method and a list of all the method arguments. You can then invoke the method on the real underlying object, or take various other actions as desired.

An Example

Here's an example of setting up a proxy:

```
import java.lang.reflect.*;
import java.util.*;
public class ProxyDemo implements InvocationHandler {

    // the underlying object that the proxy is based on private Object proxyobj;
    // create a ProxyDemo object (an invocation handler)
    private ProxyDemo(Object obj) {
        proxyobj = obj;
    }

    // create a proxy object
    public static Object makeProxy(Object obj) {
        Class cls = obj.getClass();
            return Proxy.newProxyInstance(cls.getClassLoader(),
            cls.getInterfaces(),
        new ProxyDemo(obj));
    }

    // handle interface method calls
    public Object invoke(Object proxy, Method meth, Object args[])
        throws Throwable {

        // print the method name and arguments
        System.out.println(meth);
    if (args != null) {
        System.out.print(" args: ");
        for (int i = 0; i < args.length; i++)
            System.out.print(args[i] + " ");
        System.out.println();
        }

        // invoke the method on the original object
        return meth.invoke(proxyobj, args);
    }
    public static void main(String args[]) {

        // create a proxy
        List lref = (List)makeProxy(new ArrayList());

        // create a list of Integer objects
        lref.add(new Integer(57));
        lref.add(new Integer(37));
        lref.add(new Integer(47));

        // sort the list in ascending order
        int size = lref.size();
        for (int i = 0; i < size - 1; i++) {
            for (int j = i + 1; j < size; j++) {
```

```
                Comparable obj1 =
                    (Comparable)lref.get(i);
                Comparable obj2 =
                    (Comparable)lref.get(j);
                if (obj1.compareTo(obj2) > 0) {
                    Object t = lref.get(i);
                        lref.set(i, lref.get(j));
                    lref.set(j, t);
                }
            }
        }
    }
}
```

In this example, ProxyDemo is a class that implements the InvocationHandler interface, that is, defines a method invoke() that is called when any of the methods in the interfaces implemented by the proxy are called. The ProxyDemo constructor records the underlying object that the proxy is created for.

ProxyDemo.makeProxy() is a static method that is used to create a proxy class instance, based on the set of interfaces implemented by a passed-in object.

The proxy in this example is used to track operations on an ArrayList object, an object of a class that implements the java.util.List interface. A proxy is created for an ArrayList object, and then List interface methods are called through the proxy to add elements to the list and sort the list.

When methods are called through the proxy, what actually happens is that the invoke() method in the specified invocation handler is called. It displays the method name and arguments, and then calls the actual method for the underlying object. So in this example, the proxy is used to log method calls, and then the "real" method is called. The first few lines of output from running the program are these:

```
public abstract boolean java.util.List.add(java.lang.Object)
    args: 57
public abstract boolean java.util.List.add(java.lang.Object)
    args: 37
public abstract boolean java.util.List.add(java.lang.Object)
    args: 47
public abstract int java.util.List.size()
public abstract java.lang.Object java.util.List.get(int)
    args: 0
```

The example uses java.lang.Class and java.lang.reflect.Method. These classes are part of what is called "reflection," the ability to examine and manipulate types at runtime. For example, a Method object represents a particular method of a class, and you can invoke the method by specifying an object and a set of arguments, so that:

```
meth.invoke(obj, args);
```

is equivalent to:

```
obj.meth(arg1, arg2, ...);
```

Other Uses For Proxies

We illustrated method logging above, using proxies to display each interface method just before it's invoked. You can also use proxies in other ways, for example, to add functionality to an existing interface, or to modify method arguments before calling a method.

You can also use proxies in other ways, for example, to add functionality to an existing interface, or to modify method arguments before calling a method.

using java

Life After Containers and Layout Managers: Part II

by Prithvi Rao

Prithvi Rao is the co-founder of KiwiLabs, which specializes in software engineering methodology and Java/CORBA training. He has also worked on the development of the MACH OS and a real-time version of MACH. He is an adjunct faculty at Carnegie Mellon and teaches in the Heinz School of Public Policy and Management.

<prithvi+@ux4.sp.cs.cmu.edu>

In Part I of this topic, I presented a brief background on "Containers" and "Layout Management." I also discussed the use of "Panels" and "Frames."

One motivation for this was the increased popularity of GUI-based client-side applications. For instance, in the electronic-commerce domain the canonical n-tiered architecture often relies on a "thin" client that can be a GUI-based Java program. This does not imply that the console-based paradigm is being abandoned; it suggests that using GUI applications permits a more suitable semantic format for expressing a product's capability. An example of this is the merging of two database queries from two different databases. In this example a user may prefer to click and drag icons to merge these queries as opposed to typing them at a console.

In this article, I will present the use of "Dialogs" and how to handle "Menus." This will help to consolidate the information in Part I.

Dialogs

A dialog is normally a short-term popup window with a titlebar, border, and perhaps some other pieces whose client area groups a set of User Interface (UI) controls soliciting input from a user and performing an action. The dialog is responsible for creating and positioning the controls and dealing with the notification they generate to provide a coherent dialog with the user.

In most windowing systems, because of the static nature of the control content in the dialog and the static nature of the size and position of the controls, some kind of resource (read-only) data item will describe the dialog and its content. The dialog resource can be used to easily create the dialog box and its contents and position or to resize the controls. The AWT (Abstract Windowing Toolkit) does not have resources, so dialog boxes have to be constructed dynamically in the application as required.

The AWT Dialog class is derived from the Window class. It is different from regular windows because they are dependent on a "Frame." When the Frame is destroyed, all the dialogs associated with that Frame are also destroyed. If the Frame is iconified, the Dialogs dependent upon it also disappear from the screen. When the Frame is deiconified, its dependent Dialogs return to the screen. This behavior is intrinsic to the AWT, and no specific programming is required within the application to support capability.

Recall that in the case of Applets, they are running in their own Frame. This means that they cannot use Dialogs directly. Applets must bring up dialogues in their "own" Frames.

Modal dialogs require the attention of users, and they prevent the user from doing anything else in the Dialog's applet application until it has been dealt with. By default, dialogs are nonmodal. A bug in the JDK1.1 release prevented the creation of dialog subclasses that were modal. That restriction is not there in JDK1.2.

File Dialogue

The class FileDialog is derived from the Dialog class and displays a dialog window form which the user can select a file.

The user sets up the dialog by calling methods such as:

```
FileDialog.setFile()
FileDialog.setDirectory();
FileDialog.setFilenameFilter(); (determines the initial search criteria).
```

This is an example of a "modal" dialogue. In other words, when its show() method is invoked, it blocks the rest of the application until the user has chosen a file.

To determine which file has been chosen, the application issues a call to:

```
FileDialog.getFile();
```

Menus

A menu is a hierarchical group of components. A MenuItem represents a command. A Menu (often can be a sub-menu) is a group of MenuItems or other Menus.

The top level grouping is a Menubar and contains only Menus. There is also a CheckboxMenuItem class that maintains a check of the menu item that is currently selected. None of these classes inherits from Component; they are instead a subclass of the MenuComponent class.

In order to be able to contain a MenuComponent an object must adhere to the MenuContainer interface. Recall that classes such as Frame, Menubar, and Menu are all implementations of the MenuContainer interface. Additionally these are the only classes that implement this interface. In theory, it should be possible to write a MenuContainer interface implementation to which we can attach a MenuBar, Menu, or MenuItem.

In most windowing systems, some kind of resource (read-only data item) will describe the initial state and contents; this is true even though the state of the menu and its contents may change dynamically.

The JDK 1.0 release does not support resources, so menus must be constructed "on the fly" in the code. In JDK 1.1 and subsequent releases resources are supported and this is done through the internationalizable class (which is beyond the scope of this article).

The following is an example of some code in a Frame-derived class to provide a menu:

```
MenuBar bar = new MenuBar();
Menu popup = new Menu("File", true);
MenuItem item = new MenuItem("Exit");
// set a menu bar to be added
setMenuBar(bar);
// now add component so that it can be viewed
bar.add(popup);
// now add the menuitem
popup.add(item);
popup = new Menu("Edit", true);
bar.add(popup);
popup.addSeparator();
item = new MenuItem("Cut");
popup.add(item);
item = new MenuItem("Copy");
popup.add(item);
item = new MenuItem("Paste");
popup.add(item);
item = new MenuItem("Delete");
popup.add(item);
item = new MenuItem("Select All");
popup.add(item);
```

In most windowing systems, some kind of resource will describe the initial state and contents.

For serious GUI developers, it is preferable to use the "Swing" classes because they obviate the need to implement many of the interfaces.

```
popup.addSeparator();
item = new MenuItem("Properties");
popup.add(item);
popup = new Menu("Help", true);
bar.add(popup);
item = new MenuItem("About....");
popup.add(item);
bar.setHelpMenu(popup);
```

The key to understanding the above code is that the MenuItem, Menu, CheckBoxMenuItem, and MenuBar classes all derive from the MenuComponent and *not* the Component class.

Also in order to support menus, it is necessary to implement the MenuContainer interface (Frame, MenuBar, Menu, . . .).

Finally, MenuItems generate events.

Conclusion

In Part I, I discussed the use of layout managers and their use in writing effective GUI applications in Java. I also presented an example of how to manipulate components in the absence of a layout manager at the expense of portability.

In Part II, I have presented an example of how to create a menu.

For serious GUI developers, it is preferable to use the "Swing" classes because they obviate the need to implement many of the interfaces; in other words, it is done for you. However, for those interested in understanding the details of how the AWT supports GUI development, the examples presented will facilitate that endeavor.

use your local tools

There are some times when a sysadmin has to make do with what is available. This is a story about one of those times.

by Barbara Dijker

Barbara Dijker is currently SAGE president. She's been sysadminning for about 12 years and runs a couple of ISPs.

<barb@usenix.org>

I received an email message from someone in the UK with the subject "IMPORTANT: do not ignore." The lengthy content went on and on about how they suspected that a host on my network was running a distributed-denial-of-service (DDoS) agent called Stacheldraht. Other than that, the message was strangely uninformative. It contained a link to a site with an analysis of various DDoS systems and tools to find them. But it provided no evidence of how they came to the conclusion my network was involved. Further, it said not to ask for evidence because it would be of no use.

Hmm. Was this a joke? I have to admit I haven't been keeping up with the latest script-kiddie toys. So I wasn't sure if I could have Stacheldraht in my midst and not know it. I decided rather than ignore the message, I'd at least find out if it could be true. I read up on DDoS. Stacheldraht means "barbed wire" in German. I'll let you read the full analysis at your leisure some other time at *<http://staff.washington.edu/dittrich/misc/ddos/>*. But here is the brief overview of what I learned and how I managed to find the culprit in less than ideal circumstances. I did ask the person who contacted me to at least send me the time stamps on his log entries that implicated my site. I figured I could use those to narrow down the part of the haystack I needed to look in.

Stacheldraht is based on Tribe Flood Network and incorporates features of trinoo. Agents are installed on as many hosts as possible – on hosts presumably targeted by automated probes and compromised by standard root kits. Linux and Solaris hosts tend to be the favorite targets. The agents are monitored and controlled by "handlers." In order for this to happen, the agents communicate with the handlers using ICMP echo-reply packets. Yes, just reply, no echo request. This is probably because a firewall is more likely to pass echo-reply than echo-request packets. The data portion of the ICMP packet is used to transmit control and status information. But it is encrypted. The encryption is easy to break if the agent was installed with the stock key (provided on the analysis Web site). Once installed, a Stacheldraht agent can be commanded to launch one of several popular DoS attacks: ICMP flood, SYN flood, etc. The analysis Web site also provided text strings commonly found in the program like "sickem." Stacheldraht also spoofs the source IP address of the packets it sends during an attack. If the local router will let it, it will spoof the entire IP address. Otherwise it will spoof only the last byte. It verifies this by sending a spoofed packet to its handler (with the real source address in the payload) and waiting for a confirmation.

The attack packets received by the site in the UK apparently showed my network as the source IP, but with random fourth bytes. That's why they said the log entries would be of no use. But the packets could have also listed my network while being sent from elsewhere. While that was a possibility, I figured it was still worth looking for unusual activity.

The network in question here is an ISP network. It is a /24, or "class C" network. There are over 100 hosts on that network. Each is owned and administered by a different company and person with few exceptions. I don't have user or root access on any hosts on that network except my monitoring station and my sniffer. So logging in to each system and looking for signs of compromise was not an option. That would be tedious and inefficient anyway. The monitoring I do is purely for billing. There is no firewall.

The first thing I did was check my traffic logs – one for (almost) each host. The date of the incident was a few weeks prior, so the MRTG data was reduced enough that it was

As you might expect with a large (switched) network of strange machines, bizarre things happen frequently.

difficult to see any anomalies. I found four candidate suspects with what appeared to be about 100kbps more outgoing traffic than usual. We notified their system administrator to look for possible compromise.

That produced nothing of course. They each said their system was impenetrable. They probably didn't even look. So I went back to the analysis site to look for tools. There are a few promising ones. I tried to install the two that were recommended. The problem I ran into is that they all eventually require libpcap to build custom packet headers. I don't own a Linux box. I tried a simple make where appropriate on BSDI, FreeBSD, and Solaris. All failed! I wasn't in the mood to port someone else's code. Nessus looked intriguing. I decided that would be best installed when I had the time to learn it properly – to decide an optimal and long-term configuration. So I opted to use a tool already on hand with which I'm comfortable and familiar: my sniffer.

As you might expect with a large (switched) network of strange machines, bizarre things happen frequently. My sniffer is one of my best friends. The average traffic level on this network is in excess of 5Mbps. Long ago I invested in a sniffing tool that was flexible and useful and that provides quick results. It's a dedicated beefy and zippy PC with a high-quality fast Ethernet card. I use software called Observer by Network Instruments. Alas, it has to run on NT. But it met my criteria.

The first thing I asked my sniffer to do was track and capture all ICMP echo-reply packets. In just a few minutes, I had a list sorted by number of packets. The top hosts on that list became suspect. I browsed the packet-header dumps of some of the individual packets sent and received by the top suspect. Indeed they looked like they could be Stacheldraht because of size and ICMP ID. I didn't go to the trouble of decrypting the payload to see for certain. However, I was amazed at how many of the hosts on our network send (and receive) many and frequent pings as a matter of course. That made it more difficult to use this as definitive evidence.

In the meantime, we started to suffer some minor issues on our router – it was noticeably slower than it should have been. A quick look there indicated that the packets it was routing per second increased 50-fold from typical! The problem then went away as soon as we found it, of course. So the next thing we asked the sniffer to do was to contact us when the packets per second went above a threshold. Sure enough, within a day we had a trigger. The top sender of tons of tiny packets was indeed the same host that was the champion of ICMP echo replies.

At that point we captured some of the packets this host was sending, and bingo. All the packets were TCP ACKs. They were sent repeatedly and in large quantity to a short list (<10) of destinations. Interestingly the quantity was not large enough to look unusual in our MRTG traffic graphs – because they graph bits per second, not packets per second. The source IP address used indeed had a spoofed fourth byte. We know it was spoofed because the Ethernet address was the same. (That's what we used to filter the packet capture.) We have quite a few hosts that are assigned many IP addresses for virtual Web hosts. But typically you'll only see those as the IP destination, while the source IP used is the primary IP of the host. We confirmed that the IP addresses being used to source these packets were assigned to different hosts entirely.

OK. We caught one. It was a Linux box. Then we needed to make sure our Ethernet-address database wasn't old or wrong. We unplugged its network cable to verify the packets stopped. Then we plugged it in again and verified the packets reappeared. We were confident we had the right host. Most sniffers will learn and build your Ethernet-

to-IP address table for you based on the source addresses of normal packets. You just need to make sure you do that when no one is spoofing.

We left the system unplugged and contacted the owner/administrator. They were in a meeting all day and couldn't be bothered. So we asked for root access. Our goal was to clean up the system enough so that we could bring the system back online to serve mail and Web.

We indeed found Stacheldraht. It was called /bin/in.sysched. We also found that the hacker installed their own ssh in /usr/sbin/in.amdq with its configuration files in /dev/sdc0. They installed their own "pam" authentication module in /dev/sde0. In addition, they installed their own bind (resolve.conf was still configured as a client) and their own wu-ftpd – both of which were running and presumably had back doors. Of course they replaced /bin/login and /bin/ps. Finally, some files had been made immutable so that even root could not set them back without additional effort. We instructed the customer to "nuke it from orbit" – a fresh installation was recommended.

The customer sent someone out, and they spent about five hours reinstalling the operating system from scratch, upgrading to the most recent version of their brand of Linux in the process. Within five days, they were hacked and running Stacheldraht again. The customer had rebuilt the system from scratch, but they neglected to install known patches. They thought that the latest version must be secure. Further, they never bothered to check their mail server or Web server for vulnerabilities. A quick check showed both were vulnerable.

The moral to the story? If you monitor traffic, monitor bps and pps. Have the tools you need. Use the tools you have. Know how to use them properly.

If you monitor traffic, monitor bps and pps.

the network police blotter

by Marcus J. Ranum

Marcus J. Ranum is CTO of Network Flight Recorder, Inc. He's the author of several security products and of a book on computer security (with Dan Geer and Avi Rubin) and is a part-time sysadmin.

<mjr@nfr.net>

Some Dirty Tricks

Two issues ago, I asked people to email me dirty burglar-alarm tricks from their networks. I've gotten a couple of really interesting suggestions, and I'm going to share a few of them with you. I'm editing the original emails down to a nubbin, in order to fit them in this space, so if any inaccuracies are induced, it's probably my error.

The inimitable Gene Spafford sent me a ton of good suggestions, including creating a bogus user and stuffing its /var/spool/mail file with a few fake messages from root, including discussion of various machines' root passwords. The mail files were watched by a separate process that detected when they were accessed – in those days intruders would frequently try a "grep password /var/spool/mail/*" upon entering a new machine. He also left a specialized program in the user's bin directory that looked like a setuid access program, but which, in fact, would generate an alarm and freeze a copy of the process table, etc. Another delicious dirty trick from Gene was to doctor the crypto routines in a decompiled version of the Morris Internet Worm, so that they would simply waste the bad guy's CPU cycles instead of cracking passwords. Sure enough, someone stole the doctored copy, and, presumably, enjoyed a private moment of frustration when his ill-gotten software didn't work as expected!

Daniel Wesermann described how he once rigged the routing table on a bastion host to contain a number of attractive-looking routes to nonexistent subnets, which, in fact, pointed to a "black hole" Linux machine that did nothing but suck up the traffic directed to it. He tinkered with the source to netstat to make it look like there were even active connections from within the bogus subnets, to further confuse the attackers. When a friend managed to seize control of the bastion host, and began to explore, the "black hole" machine set off alarms and logged packets.

I'm sending these guys cool "Network Police" windbreakers so they can show everyone which side they're on.

The Slaughter of the Innocents

One of the cherished ideologies of computer security in the last few years has been the notion of "full disclosure." In the full-disclosure universe, when one finds a security hole in someone else's software, one gets them to fix it by announcing the bug in a public forum (e.g., bugtraq) and including details of how the bug can be induced or exploited. Sometimes, one posts a tool that exploits the bug – as a "demonstration" – so that there can be no doubt of the bug's seriousness. The logic goes as follows:

- By disclosing bug details, we all learn how to avoid similar bugs in the future.
- By disclosing the bug's existence, we force the vendor to issue a patch.
- By releasing an exploit tool, we force the end user to install the patch.
- The bad guys may already know about the bug so we will tell the good guys.
- You/We need these tools so we can test and secure our systems.

In principle, this seems sensible, but it ignores the third parties (usually average users) who are harmed in the process of all the "improvement" that is taking place. Indeed, it's somewhat reminiscent of the old "we had to destroy the village in order to save it" kind of logic that went out of vogue in the late 1970s. Under the guise of improving security, a lot of people are being made miserable at the hands of armies of script-kiddies[1] armed with the latest hacking tools.

Let's examine the claims of the full disclosure ideologues in detail:

BY DISCLOSING BUG DETAILS WE LEARN HOW TO AVOID SIMILAR BUGS IN THE FUTURE
Unfortunately, most of the security holes that are being exploited today are categories of holes that are already well known. For example, a huge number of today's exploits are buffer-overrun/stack-smashing attacks. The Morris Internet Worm of 1988 exploited buffer overruns; we've known about that category of problem for a very long time, in Internet years. The fact that programmers keep making mistakes of that sort isn't news to anyone, either: anyone who is writing security-critical software should already be aware of array boundaries. There are, periodically, new paradigms of security flaws, but these are discovered relatively rarely.

BY DISCLOSING THE BUG'S EXISTENCE, WE FORCE THE VENDOR TO ISSUE A PATCH
Unfortunately, there are vendors that care about security, and there are vendors that do not. Vendors that care will issue patches in a timely manner, and "do the right thing" about vulnerabilities. Those that do not care will probably continue not to care. The evidence is certainly contradictory on this issue. Microsoft, for example, has pushed out patches quite quickly on some occasions, but meanwhile has allowed Windows' authentication scheme to remain so badly flawed that l0phtcrack consistently makes mincemeat of it. Email attachment execution and scripting have been responsible for several egregious security holes – resulting in band-aid patches but no fixes to address the fundamental problem. Apparently disclosure's effectiveness is mixed in this regard. I suspect that "security through public humiliation" simply annoys everyone: it insults the diligent vendor's efforts and is a flea-bite to the uncaring vendor.

BY RELEASING AN EXPLOIT TOOL, WE FORCE THE END USER TO INSTALL THE PATCH
Unfortunately, this only happens in the rare case of relatively sophisticated users who care about securing their systems. A few years ago, Dan Farmer[2] did a terrifying study that indicated a majority (60+%) of Internet sites were running Web servers with documented security holes – the end users simply had not taken the time to install the necessary patches. More important, when an exploit tool is released, it spreads into use extremely quickly. Even the sophisticated end user will not be able to monitor bugtraq 24 hours a day every day; they are still left vulnerable for an undetermined amount of time. If the proponents of full disclosure really cared to help, they would announce two or three weeks ahead of time to warn users, "on X/Y/Z we are releasing a vulnerability in PDQ that provides remote access – turn off that software immediately." Some disclosures have happened in that manner, but the majority simply appear as an announcement and an exploit tool. Hours or minutes later, innocent people around the Internet are suffering and do not know why. Hint to exploit releasers: they aren't grateful for your assistance, either.

THE BAD GUYS ALREADY KNOW ABOUT THE BUG, SO WE WILL TELL THE GOOD GUYS
Unfortunately, many of the vulnerabilities that are exposed are disclosed by their discoverers. So, perhaps the bad guys already know about the bug, but the number of bad guys knowing it is usually fairly small. Indeed, the good guys have fairly good intelligence networks of their own; usually they find out about a vulnerability fairly quickly once it leaks into the hacker[3] community-at-large. In other words, the vulnerabilities are not being exposed to the good guys much faster than they would in the course of normal events. One thing for certain, however, the vulnerabilities are being shared much more quickly among the hackers, under the guise of "helpfulness."

YOU/WE NEED THESE TOOLS SO WE CAN TEST AND SECURE OUR SYSTEMS
Unfortunately, though few like to admit it, this simply isn't the case. In virtually every case, it is possible to test for the presence of a vulnerability without having to exploit it.

Many of the vulnerabilities that are exposed are disclosed by their discoverers.

In order to make progress, we need to raise accountability for computer-security failures.

For example, rather than executing code that penetrates the system, the tool could simply notify the user that a particular piece of software needs to be upgraded. In many of the tools used by "legitimate security professionals" there are options that have no necessary purpose for *legitimate* security purposes. Take the popular `nmap` utility, for example – a tool that is useful for scanning networks to look for open ports on servers, or to identify and categorize systems – the scanner has considerable functionality that is intended to defeat firewalls and to make the scans harder to detect. Why would a legitimate security professional, acting on his own or a client's behalf, feel the need to hide an authorized scan of a network? The answer is simple: no legitimate security professional needs to act like a hacker.

I've run into security experts who claim they need penetration tools so they can convince their customers to take action. Apparently CERT alerts, vendor-issued patches and release notes, and bugtraq postings from legitimate security practitioners are not enough. I suppose it's possible someone could be so obtuse (many appear to believe the damaging effects of tobacco smoke don't apply to them, either . . .) but this is not a technical problem, it's an exercise in management and interpersonal skills. Of course, it's possible that the security expert has a vested interest in having the customer scared and feeling vulnerable – it helps sell their services.

Blaming the Victim

I've seen a distressing trend toward blaming the victim of a hacking attack: "they should have installed their vendor patches" or "it served them right for being so lame." In a few cases, when I've debated this topic over beers at a conference, I've heard even seasoned professionals display an amazing lack of sympathy for innocent victims. I think what happens is that we security practitioners are so focused on our little niche of the universe that we forget sometimes that "real users" don't want to care about this stuff – they just want to lead their lives and get useful work done on the Internet. Any of you who've been on the receiving end of a hacking attack may remember what an unpleasant violation it is, how helpless, puzzled, and frustrated it makes you feel. Not to mention how much of your time and money it eats up.

The Internet as a whole spent over *half a billion* dollars on computer security last year. This is basically money spent to accomplish no purpose other than to avoid damage. The ILOVEYOU virus and distributed denial of service attacks such as were launched at Amazon.com and eBay.com cost hundreds of millions of dollars in lost time and productivity, to say nothing of aggravation and unhappiness. A lot of the blame for this slaughter of the innocents can be laid at the feet of "legitimate security professionals" who choose to play on both sides of the fence. They want to have the secret thrill of punching weaknesses in innocent users' defenses, but they want to have their hands clean of actually pulling the trigger in person. If you read between the lines of the cases where alleged perpetrators of denial of service attacks were caught, you'll quickly see that *many of them didn't even understand how their tools worked*. They just downloaded them and made complete strangers miserable.

Accountability for All

In order to make progress, we need to raise accountability for computer-security failures. In this case, vendors that produce buggy insecure products must be held accountable if they knowingly release software that places their customers at risk. So it's not just the attack-tool writers and distributors that need to clean up their act. One of the primary intellectual underpinnings of full disclosure is the notion that vendors won't act responsibly unless they are forced – probably an accurate belief in some cases. I

don't think ad hoc force is the way to go about it and neither does the software industry. It's fairly clear that UCITA is an attempt to establish advance protection against liability for vendors producing shoddy software. If UCITA goes forward in its current incarnation then I believe it will badly hurt the cause of the end user and computer security. In moments of idle daydreaming I imagine how fun it would be to ask a judge to issue a restraining order blocking the release of some new version of a major product that was known to contain fundamental security flaws. Of course that's not going to happen, but we've got to ask ourselves what would happen to a car manufacturer that sold a car knowing it tended to catch fire and explode every so often. Eventually, if you put people at risk long enough, it will come back to haunt you. We need to make sure that happens, if we want to see change for the better.

A Plea

I beg you, please, if you find a vulnerability in someone's software, work with them to get it fixed. Don't make other people suffer for your ego by distributing something to show everyone how smart you are. If you want to show people how smart you are, write a better firewall, or a secure browser, or mailer, or something – anything – that has a useful, helpful, and legitimate purpose. Don't write and distribute denial of service tools. Don't write and distribute rootkits and trojan horses. Don't arm, aid, and abet script kiddies. If this situation does not stop, we can expect knee-jerk legislation from our elected leaders – we security practitioners need to clean house ourselves before someone else feels obligated to step in. Don't market yourself by hurting other people; market yourself by helping.

The Bottom Line

The sad reality of the full-disclosure process, as it is practiced today, is that it holds the entire Internet hostage to the views of a vocal minority that have chosen to pursue their egotistic agenda while ignoring the damage it does to innocent users. These innocent users are not technically sophisticated, do not care about security, and simply want to be left alone to lead their lives without unwarranted intrusion from a hacker. They don't upgrade their systems, they don't install firewalls, they simply do not expect that some immature script kiddie is going to amuse himself by molesting them. The immature and unskilled script kiddies are, in fact, being armed by smart but "ethically challenged" individuals who want to have their cake and eat it, too – they want the thrill of being a "black hat" hacker while commanding the salary, stock options, and peer respect accorded to true professionals. What boggles my mind is how they've managed to convince themselves they're doing us all a big favor.

Notes

1. A "script kiddie" is a hacker who doesn't really know anything about security but who knows how to download the latest attack tools from the Net and run them over and over until they find a machine they can break into and ravish. It's a term I first coined back in the early 1990s that has apparently found its way into common usage.

2. See <http://www.fish.com/survey>.

3. People have complained about my politically incorrect usage of the term "hacker" to broadly refer to cybercriminals and vandals. The original meaning of "hacker" has been usurped by the mass media, and, in the interest of being widely comprehended, I avoid arguing about vocabulary. You may feel free to mentally substitute "cracker" or "attacker" or whatever suits your fancy.

musings

More Stack-smashing Fun

by Rik Farrow

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

<rik@spirit.com>

July brought the usual stuff – crushing heat, humidity (for those not in the desert, that is), and a new technique for smashing the stack. At first, I was perplexed about why some were saying that this was not a buffer overflow, but after pouring enough water over my head to cool off and think about it, I can see why.

The problem first surfaced in reports of a root exploit of the venerable wu-ftpd server. You may recall that wu-ftpd was the victim of a buffer-overflow exploit published in February 1999. In that exploit, if the attacker could write in any directory available on the FTP server, the server could be coaxed into replacing itself with a shell, running as root, and still connected to the remote attacker using TCP.

The February exploit was a classic buffer overflow. In that attack, shell code, that is, machine instructions for the target architecture, gets copied to the stack, along with a leader of NOPs (null operations), and then many copies of an address that should point within the regions of NOPs. The idea is to replace the correct return address with the pointer to the shell code (really, the preceding leader of NOPs), so that when the function returns, it will instead execute the shell code.

There are several techniques for dealing with this. One popular one is to make the stack nonexecutable. This helps but still permits buffer overflows to succeed. The exploit must copy the shell code somewhere else, not on the stack, then overwrite the return address with the address of the shell code. When the function returns, the shell code gets executed (as it is not in a nonexecutable portion of memory).

StackGuard, subject of several USENIX papers (also, check out <*http://wirex.com*> and <*http://immunix.org*>) works by modifying the way in which functions are called. The function preamble and postamble puts a "canary" value below the return address on the stack and then checks that the canary has not been smashed on the return from the function. Now, a buffer overflow that overwrites the return address also overwrites the canary, and the StackGuard mods cause the program to exit rather than execute shell code.

The programming flaw that makes buffer overflows possible is the use of functions and loops that copy user input into a locally defined array without counting how many bytes are being copied. The C language was designed by guys who were writing an operating system (UNIX), and didn't need to worry about running off the end of an array. They wanted performance, and also knew what they were doing.

The problem occurs when a program accepts user input, whether from the command line, a network connection, or the environment, and copies it without counting to a buffer allocated on the stack (any variables declared locally, that is local to a function, get allocated on the stack). Some of these functions are: strcat(), strcpy(), sprintf(), vsprintf(), bcopy(), gets() and scanf(). You can check out the links found at <*http://www.securityportal.com/lskb/articles/kben10000082.html*> for replacements for these functions, such as strncat(), as well as some other resources for secure programming practices.

Stack Manipulation

Rather than blindly smashing the stack, the new technique enables an attacker to probe the stack, then surgically install a new return address (while not touching the canary).

Programs, like wu-ftpd, that pass user input to formatting functions, like sprintf(), are the culprits here.

The printf() family of C functions has a rather interesting capability, especially if you have learned to program using Java or SmallTalk only. That is, these functions accept a variable list of arguments. Internally, a set of routines collectively known as varargs handles the processing of function calls when the number calling arguments is not known at compile time, as with sprintf(). Let's look at a little code example posted by Pascal Bouchareine on July 18 to bugtraq (<*http://www.securityfocus.com/*>):

```
void main() {
      char tmp[512];
      char buf[512];
      while(1) {
      memset(buf, '\0', 512);
      read(0, buf, 512);
      sprintf(tmp, buf);
      printf("%s", tmp);
      }
}
```

This simple program will echo back anything that you type as input (once you compile and run it). sprintf(tmp, buf) copies the input buffer, buf, into a second buffer, tmp. The array tmp is local to main(), so it appears on the stack. So far so good.

What makes this interesting is when you include format characters in your input, such as %s, %f, or %x. To sprintf(), these appear to be commands to pop values off the stack and format them. For example, providing "%x %x %x %x" as input will result in "25207825 78252078 a782520 0" (on Intel processors and their little-endian byte ordering). "25" is the '%', "20" a space, and "78" the 'x' as hexadecimal. But just displaying what we have put on the stack is not very interesting. What if you use enough formatting commands to move up the stack until you display the return address? Now, through the user control of format commands, the state of the stack can be displayed, and the return address located.

Finding the return address is only part of the fun. There is another format command that I do not remember ever using, %n. The %n command counts the number of arguments popped off the stack by sprintf() and related functions and stores that value in the location pointed to. By arranging for %n to place values in the four bytes of the return address, you can overwrite the return address without disturbing the canary which lies below it on the stack.

With these two techniques, exploits can be written that can search for the return address, overwrite it, then execute shell code. wu-ftpd and its SITE-EXEC command logging became the target of a number of exploits all published to bugtraq within a couple of days. There was another formatting problem discovered involving setproctitle(), but no exploit for this was published. You can learn about vendor responses to this by checking out: <*http://www.cert.org/advisories/CA-2000-13.html*>.

Full Disclosure

Once upon a time, only "hackers" and a few people in universities and government research sites had access to information about security exploits. The "good guys" defended keeping this information secret by saying that the number of attacks would increase if they made what they knew public.

I am glad to have left the bad old days of keeping vulnerabilities secret behind us.

Of course, keeping secrets this way kept most sysadmins unaware of the dangers involved in not upgrading to the latest patch for service Y. (Can't use X here for obvious reasons.) The middle road involves sharing enough information to permit sysadmins to test their servers and see if they are vulnerable or not. After all, you don't want to patch something that is not broken. (You will likely break it.)

Even the middle road is dangerous, as knowing how to test for the vulnerability is two-thirds of the way to creating an exploit. But I far prefer the current state of affairs, as I prefer to know what is wrong, why it is wrong, and that it must be fixed. Also, the publishing of vulnerability information has lead to better vendor response in fixing problems.

I am glad to have left the bad old days of keeping vulnerabilities secret behind us. Now we have to deal with security problems rather than sweep them under the carpet. That is much better than living in denial.

For a free, unpaid, political diatribe visit: <*http://www.spirit.com/pol.html*>

an interview with geoff halprin

by Rob Kolstad
Editor
<kolstad@usenix.org>

[Editor's Note: *This interview was conducted electronically with SAGE board member Geoff Halprin during July 2000.*]

Rob: Tell us how education works in Australia after one graduates from high school.

Geoff: High school in Australia finishes with year 12. The exact name and nature of the final exams changes from state to state. These scores (partly exam, partly tests and assignments during the final year or two years) determine your university entrance ranking. You select your preferred courses and wait for an anxious month or two for offers.

University is anything from three years to five or six years, depending on the course. Computer science is a three-year course.

I had actually started with computers when I was 11, with PDP-8s. I was contract programming after school on Cromemcos when I was 14. So, choosing my Uni course was not really an issue. That made it all the more interesting to find all these holes in my (self-taught) knowledge when I started Uni.

I treated Uni as a place to learn (not just more exams to pass) and so enrolled in the more esoteric and challenging courses – advanced computer architecture, advanced compiler design, advanced operating systems, advanced software engineering. (Compiler design was a compulsory second-year unit that stretched everyone – we only had seven people choose the third-year elective after having to write a real compiler in second year.)

Rob: How did you prepare for your current career in sysadmin?

Geoff: I got bitten by the bug when I was young. At 12, I started attending meetings of MICOM – the Micro Computer Club of Melbourne. By my 13th birthday I had one of the first Apple IIs in the country – it ran off 110V transformers!

When I was at Uni, I was also working for Melbourne House – a computer-games company. Writing computer games (especially back in those days of 64K 6502-based and Z80-based machines like the Sinclair Spectrum and Commodore 64) was a very challenging exercise; most programming is a trade-off between speed and efficiency. Games programming requires both. You had only the "vertical retrace" (about 30–60 cycles) to update the screen, and only 64KB (minus the overhead) for the program logic, scenery databases, sprite databases, sounds, music, and work area. I learned a lot by stretching machines to their limits. On one project, we had to hand-compile C into assembler.

One of the roles I fell into there was looking after their UNIX (System III) machines. We developed cross-compilers, cross-debuggers and other tools. I was looking after those tools and the UNIX machines they ran on. I guess that was my first real stint as a UNIX systems administrator. (I'd always looked after the PDP-8s and PDP-11s, but no one knew the term "systems administrator" back then.)

So I found myself walking both sides of the fence – software development and systems administration. I had been drawn to the challenges of troubleshooting other people's code right at the beginning, as a tutor to other students, and in the support of other people's code. Managing the system was an extension of this – you had to become very adept at recognizing patterns and forming hypotheses.

Rob: Tell us how you came to be at SysAdmin after University.

Geoff: Having had my taste, I was already forming ideas about how to look after systems. I had also already had my fill of bad management. So a friend and I started a

One of the frustrations that I have faced, as a geek, is the lack of time and focus I've been able to devote to the real technical work.

company. Initially it was a shell for us to contract through. For several years I alternated between development and operations projects. I found it very useful to be able to help programmers understand the consequences of their choices (and why daily runs that take 27 hours are a bad idea).

Around 1992, after a series of such projects, I decided to step back and look at this emerging need. We were doing work with Sun in Australia, and it was clear to me that there was a huge demand for this nonprogramming skill. I started developing a product that came to be called the "operational support contract." Basically, we established good practices (our toolkit and configuration and documentation), then managed the site through regular visits, all prepaid quarterly in advance. There was incentive built in so that if we did a good job, they wouldn't need to call us, and we would have been paid a retainer that wasn't fully utilized.

We became one of two companies in Australia providing a bank of systems expertise. (It isn't quite clear who was officially first.) We had established a presence with that product, and a close working relationship with several vendors as a consequence.

I became involved in the Sun User Group of Melbourne, later to become SUG-OZ and operate nationwide. In mid-1993 SAGE-AU was founded. The groundswell had begun.

In 1995, I left that business and started The SysAdmin Group to continue my work on developing methodologies and toolkits for systems administration. It has been quite an eye opener to start up another business, and to learn what it is I do well, and what it is I rely on others to provide.

Rob: So you are the founder and sole proprietor?

Geoff: Yes. When I left my first company, the last thing I was going to do was have another partner. I had a lot of emotional turmoil to work through. So I started SysAdmin with the experience of almost ten years as a company director, but with no customer base or intellectual property – all that had to be left behind. It was a huge learning curve.

There is a book called *The E-myth*. It talks about why people start their own businesses, and the three hats you wear as you grow a business. It's quite an enlightening explanation: most people start a business for reasons like, "I can make widgets better than these guys. Why am I putting up with all this frustration and bad management?" So they go into business making widgets. And they really do build better widgets than they were previously.

But as they grow the business something happens – they have to do other things as well. There are three hats – the technician, who looks at the present ("How do I do this right now?"); the manager, who looks at the past ("How do I do that more efficiently, at lower cost?"); and the entrepreneur who looks to the future ("What else can I do?"). It is quite ironic that the first of the roles you jettison as you grow a business is that of technician – the very reason you first started the business.

The reason I say that is that one of the frustrations that I have faced, as a geek, is the lack of time and focus I've been able to devote to the real technical work. It's really quite distracting having to continually find new customers to pay the bills with. The systems administration market place has changed significantly from when I first started growing a practice in 1992.

SysAdmin has been an experiment. I've had a rare opportunity to try out different management and resourcing models, and learn the strengths and weaknesses of each.

At this point in its life, SysAdmin is a virtual company – using peers who all own their own companies as subcontractors.

Rob: What is SysAdmin's main business model?

Geoff: I have tried to align incentives for good systems-management practice with good business and customer management. For example, the OSC encourages both parties to plan, but recognizes that it isn't always possible. There are tariffs for out-of-hours work, and prepaid allotments that attract a discount but are nonrefundable. It all goes to trying to encourage good practices.

Rob: You're very active in the SAGE community. What do you think are the most important challenges that lie ahead for SAGE?

Geoff: When SAGE first began, the only people who knew what systems administration was were those in the role; the practitioners. SAGE was established with the charter of developing systems administration as a profession. I believe we are now half way there. We have established it as a unique vocation. Companies now hire systems administrators. The second half of the journey is to move from vocation to profession. This may sound quite fuzzy, and I guess it is to some extent. But there are a number of attributes that go along with the term "profession," such as accountability, continuing education, and standards of practice, to name a few.

We are facing a huge challenge. There is a drastic shortage of systems administrators, and the vendors have responded to this by putting up nice GUI screens and pretending that that is what systems administration is.

There is a drastic shortage of systems administrators, and the vendors have responded to this by putting up nice GUI screens and pretending that that is what systems administration is.

We did ourselves a huge disservice by coining the term "systems administrator." The term "administrator" implies (at least in the eyes of management) that it is a role that can be procedurized and automated. Those of us who work in the field know that there is nothing further from the truth. As the systems we integrate and manage continue to become more complex, the answers are not found in a couple of icon movements.

The word that best describes systems administration is "intricacy." The dictionary defines this as "the complex interplay of components; perplexingly entangled or involved; confusingly complex." These are all accurate representations of systems administration. We know that. The people we provide our services to do not. Just as the community appreciates the complexity of the body a doctor deals with, and the complexity of the law a lawyer deals with, so too we need them to appreciate the complex bodies of knowledge that we deal with.

For SAGE, we must embark on a major campaign of educating the users of our skills that there is more to the role than dragging a few icons across a desktop. The certification effort being undertaken at present is one of the strategic ways to address that goal. As the community sees a vendor-neutral certification program, they will start to understand that there are core skills that are intrinsic to good systems administration.

There are also new books appearing, such as Mark Burgess' book, that are addressing the disciplines and principles of systems administration. These complement the existing excellent task- and platform-oriented texts.

Rob: Which challenges are you personally attacking?

Geoff: My personal area of interest is "best practice" and standards of practice in general. I can change accountants without them having to redesign my accounts, and I can change lawyers without them having to rewrite every contract, but when I change sys-

We desperately need to develop standards of practice that we can all agree on, or at least live with as reasonable.

tems administrators, there is a huge hidden cost as they re-create the site in their own image; deploying the tools and standards they have become most comfortable with.

We desperately need to develop standards of practice that we can all agree on, or at least live with as reasonable. Even in the complexity of systems administration, there are patterns and recognized good practice. Software engineering has started recognizing the presence of patterns. Long ago (1972) they recognized the need for the "ego-less programmer." Perhaps we should learn from the world around us?

I've started (with lots of help) to define a framework that is independent of technology, vendor or platform; the Systems Administration Body of Knowledge. It has a lot of the meat still missing, but there is a structure that anyone can use to assess their site, and to plan improvement works. That's a step in the right direction.

Having a framework allows us to communicate effectively with those around us, to show them what it is that we do, how complex it really is, and to provide guidance to both personal and organizational maturity and growth.

Rob: Do you have any idea why tackling all these challenges is taking the "system administration community" such a long time?

Geoff: Systems administration is young. In the broadest sense it's only been around 40 years, but in any real sense it's been around for maybe 20 years. Accounting and medicine have been around for hundreds of years, and they're still getting it wrong.

Another major problem is that the field is not recognized as an independent field of study. A small number of universities offer units in systems administration, not a full degree, and these are part of a computer science course. Just as it took a long time for CS to break away from maths, we're now seeing similar problems of SA being treated as a subset of CS.

This is also reflected at the corporate level, where software engineering research is recognized, but systems administration research is not. It's a side effect of other research at best.

Then there's the minor problem that the industry is so massively short of capable systems administrators that the ones there are don't get much time to devote to research.

But I see this changing. The community as a whole is now recognizing the need for a more disciplined approach to systems management. This is evidenced by the outsourcing boom. I've also noticed the change in the nature of the papers at LISA over the past five years. As a group, we are now ready to be introspective enough, and have enough collective experience to recognize meaningful patterns, and to look at underlying knowledge, skills, and abilities that form the basis of systems administration. The next few years are going to be quite exciting.

ode to risk management

By Geoff Halprin

Geoff Halprin is a member of the SAGE Executive Committee, and the Principal Consultant at The SysAdmin Group, a systems administration consultancy.

<geoff@sysadmin.com.au>

It was Y2K, and a change was afoot. It was a simple change, to apply some Y2K patches to a production Solaris server. These patches had already been applied to a development server, so there was a reasonable confidence in their correctness. The change was scheduled to commence at 9:00 am on a Sunday.

The change plan, as approved, called for the mirrors to be split as a regression strategy. (All production systems are fully mirrored, using Solstice Disk Suite.) This was in addition to the cold backup that was to have been performed prior to the change.

The change commenced. The patches were applied. The machine was rebooted.

"ld.so: file not found". Oh dear. The time: 09:15.

I was called in to resolve this continuing problem at: 23:15.

The teleconference revealed that in attempting to resolve the above problem (which seemed to affect only a token ring driver), the administrator booted from CD-ROM and copied /usr/lib/ld.so from the CD-ROM to the /usr partition (A-side sub-mirror), then rebooted. Surprisingly, this did not resolve the problem.

Yes, I can hear a few of you gasping.

Once I got past the initial incredulity at the course of action, I asked whether they were aware of the consequences of this chosen course.

Did they know that the copy of /usr/lib/ld.so on the CD-ROM would almost certainly be a different binary from the patched Solaris revision, and therefore incompatible with the system? Did they know that /usr/lib/ld.so is somewhat critical to a system, and if it really couldn't be found, the entire system would not work (or boot past this error message)? Did they know how many copies of ld.so reside on a Solaris system, and where they live?

Oh yes, and that niggly one, did they know that mounting and touching one sub-mirror taints the entire mirror and effectively creates two randomly different disks which, in a round-robin read-balancing scheme (such as Disk Suite uses by default), this would cause (at best) a random panic from the kernel for reasons that would be next to impossible to trace?

Once I had teased the detail of their actions from them, I declared the change a wash, and asked them about the change plan's regression strategies. Turns out that, seeing as the change had gone so well on the test machine, they hadn't bothered to split the mirrors per the change plan, and so only had the full backup to revert to.

This was, it turns out, a blessing. The systems administrators involved had no idea of how to recover a system from split mirrors. The organisation had no "bare metal recovery procedure" for split mirrors. Some practical regression strategy that turned out to be!

So I asked them about their "full, cold" backup. It turns out, that this was an ADSM backup, performed with the system in multi-user mode (init level 3). Oh well, it's all we have, so here we go . . . . (It seemed a common ailment of that company to not understand the meaning of "cold," but that's another story.)

Now, I have nothing against ADSM as a product. I believe it to be an excellent backup solution. It is, however, inappropriate for "bare metal" recovery scenarios.

Our principal objective in making changes to a production environment is to ensure the success of those changes.

Why? Well, what is the general recovery scenario from a UFSDUMP of system partitions?

1. Boot from CD-ROM.
2. Restore to the A-side sub-mirrors (corrupted partitions only).
3. Mount the A-side sub-mirrors and remove references to mirror devices.
4. Boot the A-side (raw devices).
5. Bootstrap the mirrors from the A-side sub-mirrors.

This scenario should take around an hour or so. And from a third-party backup solution?

1. Boot from CD-ROM.
2. Suninstall a basic Solaris onto the B-side sub-mirrors.
3. Boot the B-side sub-mirrors (raw devices actually).
4. Load the ADSM client software.
5. Restore to the A-side sub-mirrors (devices).
6. Mount the A-side sub-mirrors and remove references to mirror devices.
7. Boot the A-side (raw devices).
8. Bootstrap the mirrors from the A-side sub-mirrors.

This scenario took in excess of 24 hours to complete.

So, at one level this change was a series of catastrophic mistakes and errors in judgment. But on another level, it all stems from a single fault – the lack of risk assessment and risk mitigation, skills that are essential to managing production computing environments.

Risk Management

Our principal objective in making changes to a production environment is to ensure the success of those changes – as planned, with no unanticipated side effects. They should be completed within the designated timeframe, and with no unforeseen impact on the user community.

The only way to do that is with careful planning. We are, of course, fighting against Murphy's Law. So it is important to plan for things to go wrong, things that we cannot predict. The only counter we have to an unknown threat is time – time to resolve unknown problems, and (planning for the worst case) time to admit defeat and back-out the entire change. So our change plan must include this contingency and regression time in calculating the change window required.

It is also vital that the system is never left in an indeterminate state (I hold this to be an axiom of change management), so this acts as a primary constraint on our risk-mitigation strategy selection. The state prior to the change is, by definition, a known state, with known functionality and known problems. This is why a cold backup forms a baseline for change regression, should things go awry.

Time

Assuming a stable baseline (backups), then our risks and costs are pretty much always time – time to back a change out, time to reschedule and reapply the change, lost productivity.

So our goal in risk management is to minimize these costs.

In evaluating our risk-mitigation options, we must always consider (a) how much (time) does this regression step cost to implement? (E.g., how long will the backups

take?) And (b) how much (time) does this regression plan cost to execute? (E.g., how long will the restore take?)

Splitting and recovering from split mirrors is far faster than filesystem restore from backup media (no slow tapes to deal with), and therefore a good strategy. It does, of course, come at increased risk – a disk failure during a change window would require a full restore from tape. If you want hard guarantees of return-to-service time, then a third mirror might be appropriate. Given the cost of disks and RAID units these days, doing cold backups to a mirrored disk array could be an effective alternative to tape in some environments!

Of course, copying file hierarchies sideways is an even cheaper risk-mitigation strategy (only the relevant files are copied), but not always a viable one.

Whatever regression strategy is chosen, be sure to reduce to rote procedure the steps to follow to back the change out. You do not want to have to work this out at 3:00 am under pressure of a failed change.

Further Reducing Risk

Have a good look at your change plan and identify all those steps that can be performed prior to the change window. Not only does this reduce the time required for the main change window itself, but it reduces pressure on the systems administrator, and so also reduces the risk inherent in the change.

Can steps be automated? Scripting steps ahead of a change has several advantages:

- A peer can inspect your work, and provide that essential QA review without any time pressure.
- You can create test scaffolding to find errors that the human eye is less tuned to spot.
- You greatly reduce the room for human error (such as fat-finger problems) to cause a failed change. How often have you typed in the wrong disk-partition number or cylinder number?

Geoff's Rule: Script anything where numbers or disk partitions are involved.

Of course, another aspect of reducing risk is to trial the change on a nonproduction platform. As the systems administrators in our story learned, however, this is no assurance that the production platform will behave identically to the test platform.

Problem Containment

Another important aspect of risk control is to ensure that you really do know the bounds of the exercise and can identify the point at which a problem was introduced.

Although it was never determined (don't get me started on the lack of root cause analysis), it was highly likely that the patches were nothing to do with the observed problem. The fact that the host was not rebooted prior to the application of the patches means that there was no guarantee of a stable baseline prior to the change. Just think how much time could have been saved if the host had been rebooted prior to the change, and this problem encountered at that time?

Any nontrivial change plan should have regular checkpoints scattered throughout it (including at the beginning). These are points at which correct system behavior can be confirmed, and thus problems detected early, and attributed to their cause faster and more accurately. The earlier we detect a problem, the better position we are in to take corrective action and continue with the change. By contrast, if we do not detect the

The earlier we detect a problem, the better position we are in to take corrective action and continue with the change.

Systems administrators must understand the risks inherent in the changes they are performing and plan for things to go wrong.

problem until the acceptance tests at the conclusion of the change, then our only course of action is to enter a protracted troubleshooting exercise, or back the entire change out.

When is Cold Not Really Cold?

Finally, another hard-learned lesson. I always thought that what constituted "cold" was pretty obvious. It turns out that not everyone shares this insight.

A "cold" backup (as opposed to a "warm" or "hot" backup) means a known, stable point in the system being backed up. We should always be able to recover from a cold backup and know that the system will operate properly, and the integrity of the data is assured.

The only way to perform a cold backup is with the relevant object (e.g., database, filesystem) quiescent, with no changes occurring. This is why cold backups are normally performed in single-user mode on UNIX.

I have seen people performing "cold" backups in multi-user mode (run level 3), and with a database live!

It is important to note that even with no users on a system, you are not guaranteed a stable cold snapshot if you do not ensure that the host is in the appropriate run mode, or that other steps are taken to guarantee the machine (or relevant subsystem) is quiescent. For example, CRON can kick off jobs that alter the filesystem during a backup, and upstream applications can FTP new transactions onto the host. Should either of these happen during a "cold" backup, we have created the potential for data loss.

Conclusion

Change management is a critical skill for systems administrators, whether you are running systems for a Fortune 500 company, an e-business that is exploding in capacity every week, or even a small business with relatively few systems to manage. Businesses rely on computers as mission-critical resources. It is our job as systems administrators to minimize service disruption. Change management is a core strategy and discipline for achieving that end.

A core element of change management is risk management. No change should happen to a production environment without appropriate risk assessment and planning. The weight of such an assessment should be proportional to the cost of the change failing and the time required to take corrective actions.

Systems administrators must understand the risks inherent in the changes they are performing and plan for things to go wrong. The same skills that we use in reactively troubleshooting system failures should be employed in proactively planning system changes.

vendor contract services

A Primer

by Strata Rose Chalup

Founder, VirtualNet; Consulting Director of Network Operations, KnowNow Inc. Strata Rose Chalup got her first sysadmin job 1983 and never looked back.

<strata@virtual.net>

I'd like to talk about the logistics of coordinating large-scale service build-outs when you have several independent vendor teams and subcontractors working on a project. More and more of the "architecture" work I do is devolving into 30% architecture, 50% project management, and 20% keeping the finger-pointing and name-calling between vendor teams at a minimum. I know that I'm not the only one in this situation, and I'd like to share some of the things I have learned along the way.

I have been both an employee and a contractor for clients, and a subcontractor for a vendor team. I've managed projects involving vendor teams, managed vendor teams for a client, and managed vendor teams for the vendor. In other words, I have toured the sausage factory from most conceivable viewpoints, including that of the sausage. I won't say that I will never eat another sausage, but I will say that "managed eating" is a Very Good Thing when it comes to sausage.

I will also say that while there are few "right answers," some of the more correct answers from a client side run headlong into best practices from a vendor side, and vice versa. You may be frustrated at the lack of a clear recommendation as to the best course of action, as am I. The best that one can do in this situation is to illuminate some of the issues and tradeoffs, and allow each person to make the choices that seem best for their situation.

Vendor 101

THE STATEMENT OF WORK

First, understand that vendor services groups, unlike many individual contractors, generally have a well-defined process for engaging in work. They may only deal with firms as part of a product sales process, or they may deal with firms that have purchased products in the past. They tend to require a signed and approved Statement of Work (SOW) before letting the client talk to anyone but a sales person. The SOW is basically a contract defining the client and the vendor obligations. A good SOW will protect you, the client, with an appropriate level of detail about deliverables and about the process whereby deliverables are approved. Since the SOW must be approved before the work begins, and contains the financial information controlling the work, getting a good SOW in reasonable time is something of an exercise in compromise. Specifics will be detailed in the next section of this article.

Be aware that your vendor may look like a large, homogenous, well-managed corporation, but in fact may be a tangle of different subcompanies that have been acquired, autonomous departments, software/hardware organizations with completely different tech support and vendor services groups, etc.

To make things more interesting, even within one company or group, a vendor may have two distinctly different vendor services departments. One will be a sales department that books an SOW, and the other will be a deployment department that is basically stuck trying to deliver what the sales folks have promised. If your vendor sells hardware as well as software, it is extremely likely that there will be *at least* two different vendor service groups, hardware and software, which may also contain this dichotomy within them. Just because you have signed on an SOW does not mean that the vendor

actually has personnel available to assign to your work. It is generally a true sales process, meaning that the goal is to get you to sign first and then to deliver afterwards.

Most vendor services groups I have seen seem to be constantly shifting between a one-structure and a two-structure arrangement. The truth of the matter is that vendor services are hard to deliver, since you are dependent on the usual intricacies and delays of the sales process plus the real-world constraints of personnel availability. There is no "perfect fit."

THE VENDOR TEAM

Many firms are in a constant battle to retain personnel and to protect them from burnout. Being a vendor services team member generally means experiencing most of the angst of consulting (variable schedule, constant emergencies, frequent travel) without the benefits of consulting (high rates, vacations between contracts, freedom to turn down work). It doesn't take a rocket scientist to figure out that a move into actual consulting or into another department or company can result in higher compensation, reduced job pressure, or both.

Your vendor team will probably be composed of vendor employees and of contractors. Either the employees or the contractors could be people who have worked extensively with the products, or people who are new to the vendor and whose resumes indicated product familiarity. If the team member is new to the vendor, he or she may not have the crucial engineering and support contacts needed to resolve problems with your deployment. As a contractor myself, I do not wish to tar vendor services contractors with any sort of broad brush. Most of the ones I have worked with have been talented professionals who have conducted themselves admirably at client sites. In practice, however, it is more likely that the vendor services employee has established relationships with key engineers and high-tier support personnel. He or she can bypass cumbersome tech-support protocols with private email to internal developers, search the employee bug database, etc. It is a very good idea to make sure that for each major vendor product in your deployment plan there is one person, employee or contractor, who has a strong history of working tightly with the vendor on that particular product.

As I have often discovered, the right hand rarely shares resources with the left. A truly dynamite vendor team member who has access into a vendor's VPN product may have little or no access to the same vendor's firewall product group. Familiarity with the engineering group responsible for the messaging product in no way guarantees knowledge of the engineering group responsible for the Web-server product, etc. This is of particular importance when dealing with a multi-product deployment from a large vendor. Some of the products were probably developed in-house, some may have been acquired as intellectual property, and some may have been acquired along with the company that developed them. You have no visibility into the internal support structure of the vendor, so you must ensure that the expertise will be available to you via your vendor team regardless of the strange politics that may hold sway behind the scenes.

Role	Hourly Rate	Duration
Project Manager	$250	12-14 weeks
Messaging Architect	$225	2-4 weeks
Messaging Developer	$175	4-8 weeks
Scripting Developer	$175	2-4 weeks
Firewall Architect	$200	1 - 2 weeks

Table 1: Sample project assignments

DEPLOYMENT AND FINANCIALS

In the vast majority of cases, deployment of the vendor team will be assigned and specified by role and duration. Where this is done, rates will be assigned via roles, rather than individuals. An example is given in Table 1.

Given personnel changes and project constraints, it is very unusual for a vendor services organization to list specific individuals as committed to a project in the SOW. If your firm has experience with any specific individuals and wants to require that they be part of the team, it is necessary to capture that quite explicitly in the SOW or in an addendum that is indicated as binding by the SOW. Issues surrounding role-based resources vs. individuals are discussed more thoroughly in the "Resources" section below. The Statement of Work will generally require a financial commitment for the maximum estimated time listed for a given role. Be certain that your vendor's SOW indicates that only the actual time worked will be billed, rather than the entire allocated duration of the role.

An overhead figure will be set in the SOW for incidental vendor expenses, including reimbursement for travel expenses for personnel who are brought in from out of town for the duration of their role in the project. Typical numbers are 12% to 15% for overhead. Note that if your firm is located in an extremely expensive urban area such as San Francisco or New York City, this figure may be higher. You may be familiar with "overhead" from a university or research environment, where it is deducted by your parent organization from the total monies allocated. The type of overhead expense mentioned here is only billed against to provide direct reimbursement for expenses incurred by the vendor on the project.

That said, it is up to your firm to track these expenditures closely and ensure that they do not exceed the figures budgeted in the SOW. The vendor organization does not typically track expenses vs. overhead allotment, relying on the client to protest if the expenses exceed the allotment. It is often helpful to request a copy of the vendor's travel and expense policy for employees and subcontractors. The vendors generally review individual team-member expense reports against the criteria in the policy.

Be aware that it is very common for vendors to pull in employees and subcontractors from other regions in order to fill out a team. Most projects have a rate structure such that the vendor can profitably employ subcontractors from other regions as long as there is an overhead for travel expenses paid by the client. Do not assume that the overhead will go unused if the vendor is local to your area or has a regional office nearby.

Client Requirements

Know Your Own Constraints

First, ask yourself the Very Hard Questions: Can anyone really get this project delivered in that timeframe? Do you have any idea of the complexity of what you're doing? Have you read *The Mythical Man-Month*?

We can talk about that in another article, but for now we'll stick to the rack-space rental, bringing in your WAN lines, setting site policy, hiring your ops staff, setting up your trouble-ticket system, setting up your tier-1 support including 800-number help desk, etc. Okay, maybe your project is different: small, manageable. Probably not, since companies never seem to call vendor contract services for those kind of projects – they just overload their sysadmins. Or worse, they may not even have enough of a realistic schedule to get beyond the "I don't care what it takes, just do it" mentality. Marketing has been planning this for a year, engineering has been writing the custom app, and now, six months before launch, the exec-level project plan says "hire a director of oper-

First, ask yourself the Very Hard Questions: Can anyone really get this project delivered in that timeframe?

You should have a distinct plan for what you wish the vendor team to accomplish.

ations and commence buildout of site." You have my deepest sympathies. Exec staff always seems to believe that a big enough budget can change the law of physics.

Are you doing this to spare your folks the ramp-up time on getting things working, or are you secretly hoping to offload a whole bunch of stuff onto the vendor team and use them as generic sysadmin help? Are you really prepared to give them a *working* infrastructure or are you building everything in parallel and wasting a lot of their time (which you will pay for!) while you fumble with your network plan?

Define Your Goals

You should have a distinct plan for what you wish the vendor team to accomplish.

Note that if what you are trying to do is very complicated, you may find limitations in the products or their interactions of which you were formerly unaware. For this reason, I recommend purchasing the minimum number of licenses or copies of the software required to establish a working test setup. If you are integrating a number of different vendor products, you will undoubtedly find limitations in their interaction that will require additional engineering work on your part. You may even find limitations that call the validity of your entire production model into question.

A typical e-commerce project may be attempting to integrate new accounts, provisioning existing/new accounts, billing, Web forms, messaging, security via certificates, advertising, and a store or sales engine. Companies or products involved might include PeopleSoft, Oracle, Oracle Financials, custom scripting, LDAP servers, LDAP gateways between PeopleSoft or Oracle and the LDAP servers, messaging servers, mailbox provisioning with "welcome to this" messages, certificate initializing, setting uniform passwords across services, etc. There are so many places for things to go wrong that it boggles the mind.

If you have an unusually strong requirements document and someone to drive the project to the level of detail necessary, you have an excellent chance of finding out what works and doesn't work during the design and architecture stage. If not, you will have to find out in reactive mode while trying to meet your deadlines, and go through annoying and probably expensive change control with your original SOWs and project plans while building the live system.

Constructing the Statement of Work

Project Plan vs. SOW

To build an SOW, you should have real deployment plans, with clearcut deliverables for your team as well as for the vendor team. The usual catch-22 is that you generally need the assistance of the vendor team to make those deployment plans. It would be wonderful to have an actual project plan delivered as part of the SOW, especially for these big $250K-to-$500K projects that involve a whole vendor services team. This seems eminently sensible from the client viewpoint. Your vendor manager will balk at this, since developing project plans is usually a source of revenue and part of the service provided. If you ask for a project plan as part of the SOW, you are asking him or her to commit an expense for "business development" to create that project plan, i.e. to adversely affect his or her bottom-line numbers.

You could attempt to insist that the project plan be listed as part of the SOW, but delivered *before* the SOW is approved. You will not get this kind of concession from an experienced manager, as you are asking him or her to commit resources before any legal guarantee of payment is in place. This is reasonable, as lack of an approved SOW also

exposes your firm to the risk that personnel will be needed elsewhere and reassigned, leaving you in the lurch.

In theory, it would be better to insist on separate SOWs for the deployment plan and the "real work," so that the mutually agreed-upon deployment plan can serve as the basis for estimations. In practice, this means committing to an architecture SOW separately from a deployment SOW.

Be aware that without a detailed project plan, the vendor account manager will be unable to get accurate estimates from the vendor architects and potential team members. This could lead to a highly inflated fixed price, or the imposition of time and materials with an unreasonable cutoff point.

Multi-stage vs. Monolithic

Clearly we are back to our catch-22. I still believe that the best way to proceed is to arrange a fixed hourly allotment for preparation and review of a project plan, including identification of resources required by the vendor team. This "mini-SOW" will get things off on the right foot by getting you a firm set of plans and deliverables. It will almost certainly need to include architectural design and review unless you are merely hiring vendor implementation for a predefined architecture. Most contract services groups will pressure you to do one big SOW that includes architecture and deployment, but I caution you against this. While the vendor may present some compelling arguments, most of them boil down to "I really want to book this." Be aware that the manager who books your SOW receives a very substantial commission bonus in many, if not most, vendor contract services organizations. In a split sales/deployment contract services organization, the manager may even be subject to quotas or deadlines similar to those in the main product-sales group.

Be aware that approval of an SOW on the vendor side may be a one-to-three-week process, as it has elements of a binding contract and must be run past the vendor's legal department. A "dead zone" of one to three weeks between the project-plan phase and the deployment phase may affect the availability of the vendor team, since there are usually more projects than personnel to implement them. Creating the larger SOW in parallel with the mini-SOW, incorporating its deployment plans as they come in, is the ideal approach. Tip: If you and your vendor manager pre-approve the wording of the SOW shell first, then add specific deliverables, you can shave days or weeks off the approval process when you have your final large SOW draft ready.

You can usually get a good feel for the availability of team members from the account manager, and should plan multiple vs. multi-stage SOW from there. If a vendor manager claims that he or she cannot guarantee that you will get the resources that you need unless you commit now and schedule their time out N weeks, I would think seriously about another vendor. What that manager is really telling you is that they have a very disorganized or loosely organized department that is long on managers and short on technical talent. In those environments, managers with paying work get to reserve architect and senior-class people, usually through some project-tracking system.

If your choice of vendor is dictated by the product(s) you have purchased, you are between a classic rock and a hard place, and you need to decide for yourself how much is bluff and how much is real. Attempting to close quickly on a "mini-SOW" is probably the right choice, since your vendor manager is also looking at lost revenue if he or she can't pull a team together.

If a vendor manager claims that he or she cannot guarantee that you will get the resources that you need unless you commit now and schedule their time out N weeks, I would think seriously about another vendor.

Resources

Make sure you get commitments of specific personnel with specific experience, not "or equivalent." All vendor services organizations are subject to pressures that encourage them to pull architects and senior folks off and put you in a sustaining role before you are really ready. You *want* those senior-level people around when you start testing, for instance, since they can debug problems much faster than the junior folks. And when the SOW says "equivalent," it means you'll be paying the same dollar for those junior folks as you would have if you'd stuck to your guns and kept your original team. Watch out for this one, it is a big one and quite common. Just smile sweetly and say that if so-and-so is an "equivalent resource" then he or she can go off to the East Coast to do the work that your original architect is supposedly called away to perform. After all, they're "equivalent" aren't they?

There are some legitimate situations in which you may need to okay resource substitution for a period of time. Perhaps your senior implementor came to your project from a six-month gig at BigCompany and they have run into a problem. You can understand that he or she could solve the problem faster than an equivalent, due to the prior familiarity with the site. You hope that if you run into problems down the road that you would be able to command the same priority on a few days of the person's time. That's fine, and is part of the inevitable give and take. It's when the vendor wants to start a new project with your paid resource that you should be prepared to hunker down and not give in. If you do agree to a substitution, specify a maximum number of days, after which you must reapprove the substitution in order for the vendor to be meeting the contract terms.

Part of the SOW should include measurable, quantifiable milestones and checklists for determining when those goals are met, so neither you nor vendor team will say "but it isn't done." If those checklists are not available at the time of the SOW, you must make them an explicit deliverable with a signoff stage. You must also be willing to acknowledge any of your own mistakes, and pay for them if necessary. An example from real life: the network is stable for several weeks, then the client decides to rip out the core router and rewire the racks *after* the vendor team has conducted performance testing. Are you sure the new configuration is stable enough to let the old performance testing stand? What if the new load balancers you are introducing don't really do a valid RSET on expired cached connections, and thus confuse the LDAP connection caching on the mail relays? Are you going to find out during your beta test phase that if no email passes through the system for five minutes or more, the mail relays get into a wonky state? A typical true, if ugly, story.

Documentation

One common "gotcha" is documentation. The SOW should specify not only documentation on any software installed by the team, but also documentation of config files. These should include annotations and/or explanations in or with the file, not just printouts of the file contents. It should also be explicitly stated that any scripts, programs, or utilities brought onsite by the vendor team to do their job should remain installed onsite *and* be documented by the vendor team, even if those tools are not official products of the vendor. I would advise that the wording indicate that tools explicitly published/provided by outside groups such as the Free Software Foundation are exempted from this requirement, as are tools published by individuals outside the vendor group. Any modifications to those outside tools done by vendor group employees or contractors, however, even if performed outside the duration or scope of your project, should be required to be documented.

Outside Tools

To protect yourself, I also advise inserting a clause that indicates that any licensing or usage fees, copyright issues, or intellectual-property issues involving tools, scripts, or utilities brought onsite or requested/required by the vendor team are the responsibility of the vendor and that you will not be liable. If that nifty set of host file to LDIF scripts turn out to require an annual fee if used commercially, for instance, your vendor should not be able to say that those scripts are crucial to their completion of the SOW and that you'll have to pony up. If they say that *before* the SOW, that's fine, at that point you and they can negotiate for the extra time and effort needed to work around or duplicate that functionality. What you don't want is to find out halfway through the project that you're in violation of someone else's shareware licensing.

What you don't want is to find out halfway through the project that you're in violation of someone else's shareware licensing.

Controlling Scope

Note that your vendor services account manager is almost certainly subject to "variable compensation," i.e., a commission. He or she will be genuinely helpful, but also attempt to sell as large a contract as possible. Be realistic about what portions of the job can and will be done by your staff, and what portions should be implemented by the vendor team. If you assign too much to your team, you will be paying for the vendor team to twiddle their thumbs while your team struggles to complete dependencies so that the project can move forward.

One of the most helpful practices in this area is to encapsulate functional pieces of the project such that you minimize interdependencies but allow both teams to proceed in parallel. An example of this would be specifying that your team will be providing the machine infrastructure and firewall, whereas the vendor team will be installing and configuring their application servers on the machines using the assigned names and IP addresses, then cooperating with your team to assist in the final testing of the firewall.

It is vital to make certain that the SOW explicitly includes all stages of the product-deployment lifecycle, since the vendor team will be using the SOW to determine their duties. Many items are commonly forgotten or merely assumed by the client, and there is friction introduced when differing expectations collide. Do you have a test lab? Will the vendor team be installing the products on the test lab first, testing them, then installing them on production hardware and re-testing? Is the testing simple functionality testing, or is it performance testing as well? What are success characteristics for the installation, and are they functionality-based, performance-based, or both?

Recognize that most vendor services teams' first goal is to get the job done. Their very close, but still second place, goal is to make you happy. Question, question, question, and then question some more. Make sure your site practices and standards are fully captured in the SOW, so you don't end up with a vendor team insisting that it's your responsibility, for instance, to make Jumpstart scripts for their packages since the SOW only says "configure" and doesn't spell out "provide working Jumpstart configurations."

Review SOW architectures with the vendor team leaders in a wider sysadmin context. Many of the architecture-level vendor team leads I have met are actually fairly recent CS grads who are very sharp in their vendor's product but have little or no clue about sysadmin "best practices." They tend to produce vendor architectures that are excellent from a high transactions-per-second throughput-rate perspective but poor from a redundancy, maintainability, and disaster-recovery perspective. For example, little or no attention is generally paid to preventive-maintenance issues such as alert levels, log rotation, etc, leaving your team to reconfigure (and possibly break) the vendor packages after the vendor team has signed off on them.

Working Together

Provide Appropriate Infrastructure

Vendor rates and time estimates do not generally include spending half a day here and there getting FTP to work between your colo hosts and your internal network, for instance. It's up to you to have those things in place. It's perfectly legitimate to ask the vendor team to provide a list of what facilities they need upfront, though, and to insist that the list be meaningful rather than vague. A typical response might be "normal system functions available," to which you might push back with a list of services and checkboxes as to which are truly required. Be aware that many vendor teams rely on Internet access to pull binaries and patches from their own sites, so be certain your list specifies "internal" vs. "external" access for services such as Telnet, ssh, FTP, and email.

A savvy vendor services team will come with a project manager, who will insist that any delays caused by your infrastructure be included quite explicitly in the project plan, and who will bring change control processes to your management if presented with a non-working environment. Be aware that multiple "little things" that seem inconsequential to you will add up to significant time spent wrestling with your environment. If you succeed in obfuscating the issue, all you are really doing is delaying the inevitable, and pushing the argument off to the sign-off portion of the project. If the vendor is doing time and materials, the higher cost will come out in the end. If there is a fixed-price contract in place, certain items like documentation and testing phases may end up being silently ignored or handwaved around because of the extra time taken previously. Yes, it is unprofessional. Yes, it happens all the time, especially once the servers are working and your management has lost interest in chasing the vendor for some missing docs.

Team Interactions

It's a good idea to hold cross-vendor "global group" meetings, with your own people and the vendor team. Obviously if you have all of the vendor team members in your weekly staff meeting, you are paying a lot for them to sit around and eat donuts with your team. Structure regular attendance at your staff meeting for their project manager and technical lead.

Don't let your team drag the vendor team into kibitzing on things that aren't their SOW. Many of the folks on the vendor team will have relevant experience that could benefit your group, but you must be absolutely clear to both your team and the vendor team that they are hands-off! Be certain that the vendor team manager is very clear on your policy, as he or she may need to remind individual team members that they are not allowed even to "take just a quick look" at something outside the scope of the work they are doing.

Do plan for your team to spend some unstructured time, such as a team lunch or coffee break once or twice a month, with the full vendor team to get the benefit of their experiences on projects like yours. An hour of trading war stories not only builds rapport but gives your team an idea of who actually knows their stuff on the vendor team and who is just full of it – and vice versa! Some of your own team may not be as strong as you'd like, and a good relationship with vendor project managers or technical leads can often yield information about your own people as well.

Brief your team and your support organization about the vendor team's existence, their project, and its priority in the scheme of things. If you are the member of a networking group in a large company, working with a switching vendor on your e-commerce site,

it's likely that the folks at "support@mycompany" have never heard of the vendor team and will sit on their requests for a couple of days trying to figure out who they are and to which department they report.

Unofficially Managing Vendor Teams

You may or may not be your firm's client contact for the vendor team. You may be a technical lead who is working closely with them. Or you may be a manager to whom the vendor team reports as a whole, but individual vendor team members report to their own project manager. Regardless, there are things you can and should do to "manage" the vendor team and provide some guidance.

The first thing is simply "be nice to them." Yes, you've got better things to do than coddle other people's employees that you're hiring at $150+/hr. Usually, so do *their* managers. Most professional-services engineers are overworked, underappreciated, and on their own with little management support. Their managers are usually out lining up gigs for them that demand weekend travel or 24x7 on-site presence. If you treat these folks more like your own team than strangers, often they will jump through hoops for you and that may make the critical difference for your project.

You can provide some extra support, give positive feedback whenever appropriate and be considerate in giving negative/constructive feedback, include them in some of the lunch runs and T-shirt giveaways, etc. They will usually take an interest in your project and begin to think of you as a friend and colleague more than just another client to survive. Miracles can and do happen, ranging from custom features finding their way into a release to client project staff being put on the "friends" list for an IPO. (That last is a true story from a recent project. I was on another vendor team, alas, and arrived on-project very late in the game. Oh well! Maybe next time!)

On the flip side, prepare to spend what seems like endless and needless times nudging vendor team members through fairly simple roadblocks. They are usually operating under stress, so often their tendency is to want to fingerpoint, quit for the day, and go back to the home office to catch up on paperwork, email, etc. It's not malicious, it's simply a response to things that seem to be out of their control, so they are going to exercise good time-management skills and do something else.

Make sure that someone from your team is always available to the vendor team so that he or she can catch these things in action and insist on trying to walk through the obstruction, fix the problem right then and there, make internal requests within your organization, and generally grease the wheels. You are probably on a tight schedule, and you'd rather have one of your team members call the Security group and arrange a quick chat with the firewalls guy than have your vendor team stuck for half a day unable to test sending Internet mail. In most cases, it is better to appoint a dedicated "fixer" as liaison for the vendor team than to have them try to resolve support issues directly with your company's usual channels. At the very least, the "fixer" can lodge support requests on the team's behalf, and may be able to provide direct results if the project is time-critical.

On-site vs. Working from Vendor Offices

There is one policy which you should absolutely attempt to enforce which is guaranteed to be unpopular with the vendor. That is the insistence that work be performed on-site, and that assigned vendor team members will be required to work on-site unless specifically exempted by mutual agreement. Obviously you are not going to insist that sick days, etc. be verified with your own firm's manager, but you should make it clear to

Make sure that someone from your team is always available to the vendor team so that he or she can . . . generally grease the wheels.

Insist on regular reports from the vendor team.

the vendor manager or team lead that any absences need to be reported and also scheduled in advance if at all possible. That means the weekly report should include upcoming absences such as holidays, scheduled vacations, etc. of the vendor team members. If a team member has an unscheduled absence, it should be reported via email from the vendor manager or team lead on the same day.

This may seem like pure old-fashioned control-freak business thinking. Unfortunately, speaking as someone who has worked on both sides of the fence, I find that it is a key factor in a successful outcome when working with vendor teams. While vendor professional-services groups look great on paper, the fact is that the majority of them are extremely short-handed. When working out of their normal offices, vendor team members are subject to phone calls about past projects, visits by sales managers trying to get help closing deals with "just a quick conference call," deployment managers asking about availability for the next project, fellow team members with a problem that needs solving on another project, etc. Of course, all of these people have pagers and cellphones, but the degree to which they respond to non-urgent messages delivered remotely is vastly different from the response to someone "just dropping by."

I have seen most of a day be spent handling other projects' needs or administrative issues, and the norm is generally two to three hours out of an eight-hour workday. Many vendor team members are very conscientious and will not charge those hours to your project. I have seen many others who will just cheerfully write down "8 hours" for that day, even though most of them were not hours that assisted your project in any way. If the team member is an employee of the vendor, he or she is probably subject to a performance criteria that includes 35–40 hours of billable (revenue) time per week. If he or she is a subcontracted consultant, he or she is doing vendor-related work during those interruptions and certainly has little motivation to donate hours to your firm.

The flip side of this issue is that there is generally great utility in having one or more of your team members working out of their home office for a half-day or a day per week. They will be meeting with coworkers to discuss solving your problems, going down the hall to visit engineering with questions, and basically interrupting other people to help your project. Yes, I understand that is slightly hypocritical. So on the basic theory of commutative benefit in a business situation, it is in your interest to let folks leave after lunchtime one day a week to work from the home office or designate a day that they will always be working there. I do stress that you will almost certainly lose control of the project if you allow the vendor team to set its own hours and on-site policies, as they will choose to work remotely most of the time and communicate solely through architecture documents and occasional emails.

Status Reporting

Insist on regular reports from the vendor team lead or project manager, as well as brief status reports for the global group meetings. The weekly status reports do not need to be lengthy, but they should follow a standardized format which includes the following:

- Reporting period
- Issues hotlist
- Resolution of last report's issues
- Team members on project
- Hours billed to SOW
- High-level goals accomplished during this period
- Planned work done during period (by team member)
- Unplanned work done during period (by team member)

- Goals for next period
- Change control information (cumulative, but old info can be pruned if it exceeds a page)
- Updated copy of any project plans or timelines maintained by vendor team
- Scheduled absences (by team member)

Multiple Vendor Teams

If you are implementing one of the complex service clusters alluded to in the beginning of this article, it is very likely that you are dealing with multiple vendor teams, ranging from your layer 2/3 switching and load-balancing vendor to your layered application-server-package vendor and everyone in between. I feel for you. I truly do.

In addition to the tips and techniques listed here, I strongly recommend that you arrange regular team meetings among the multiple vendor teams, with the technical lead for each major product and the vendor team project manager from each team participating. This works best if you can get overall authority for facilitating, but also works if your boss is a good facilitator. These people may be attending a regular staff or subgroup meeting with your team anyway, but that is not a forum where you can drill deep on product interactions.

Be sure to build verifiable, quantitative functionality test suites into the SOW as a deliverable for each vendor team. When problems arise, insist that each team repeat the simple functionality table and show results. Don't take "we didn't change anything, it should all still work" for an answer.

I know of one major vendor who began building its own vendor services team last year simply because it was losing too much engineering time resolving finger-pointing by other vendor services teams. Since this firm had no team of its own, there was no one on-site to point back and issues were escalating needlessly into engineering to prove that its product was not at fault before other vendor's teams would initiate diagnostics. Be ruthless about the teams doing diagnostics and functionality tests in parallel when the project hits a major snag. You are the client. You are paying for this, and it needs to be resolved.

Conclusion

A complex service rollout involving one or more vendor services teams is usually a wild ride, but it can be one of the most exhilarating and satisfying experiences found in the workplace. Rigorous application of some common-sense rules can result in a project that ends with everyone feeling good about the project and looking forward to working with one another again someday. Letting things "just happen" and assuming that things will proceed as well as they do in ordinary projects is almost a guarantee of problems somewhere along the line. With knowledge of the workings of the vendor services process and some preparedness in setting up the engagement, you can greatly increase your chances of a fully successful outcome.

Letting things "just happen" and assuming that things will proceed as well as they do in ordinary projects is almost a guarantee of problems.

finding time to do it all

by David K. Z. Harris

David K. Z. Harris has been a network plumber "for more than a decade." He's been a member of the technical staff at Certainty Solutions for nearly three years.

<zonker@certaintysolutions.com>

and Bryan Stansell

Bryan Stansell is one of the earliest members of Certainty Solutions staff, and is the current keeper of the Conserver code.

<bryan@certaintysolutions.com>

David Stuit, and Michael Batchelder, both of GNAC, also provided substantial material for this article. (In August 2000, Global Networking and Computing (GNAC) became Certainty Solutions.)

During the past few years, we have seen an explosive growth in the number of companies that depend on the Internet for the success of their business. For those of us who work to support the systems and networks, we can look forward to stable employment. But the success of our companies means more work to do, and it is becoming harder to find additional qualified technical people to hire. As a result, we need to learn to do more with less. This article will tell you about one tool that saves me time, reduces downtime, speeds troubleshooting, and aids in the training of new staff. (Oh yeah, it's also free!)

Thomas A. Fine and Steven Romig presented a client-server application called Conserver at LISA IV (Fall 1990[1]). Conserver allowed administrators around the network to have access to serial-console ports using the Telnet protocol between the client and server, and the server would log all of the session activity coming from each console. This application saved Ohio State University space, power, and cooling by removing many of the monitors and keyboards that used to be attached to the servers in their data center. It also saved them a great deal of running to the server room every time they needed to make a change on the physical console! I'm happy to report that Conserver has been getting better over time.

During the past decade, the Conserver application has evolved[2], based on the needs and feedback of the users. Terminal servers, as a product class, have also evolved during this decade, and many now allow Telnet-like (socket-based) and secure-shell (ssh) access from networked hosts to individual serial ports. This extends the reach of the administrators to the far corners of their network, including control of devices that do not have network connections (such as CSU/DSU equipment, test and diagnostic devices, to name only a few).

Conserver is distributed freely[3], but the serial ports you use with it, unfortunately, are not. While a terminal server can provide remote access to serial-console ports, the cost-per-port to deploy them has been high, and is currently still higher than the cost of a 10/100 switch port. There are many advantages to be gained by deploying terminal servers in your network, and I will outline some of the best reasons in this article. I have found the benefits well worth the cost of deployment, especially in terms of the speed of recovery from outages. And in today's e-commerce world, some outages can be downright costly, in terms of lost sales, not to mention customer perception.

When your hosts won't boot, access to the serial-console port becomes invaluable, but the serial console can also be useful in your change-control procedures. Once your hosts are up and running, the administrator(s) will have many options for remote access to them to control their functions. But those avenues of access can mean that multiple administrators (or users with administrative access) will have simultaneous ability to the change settings on the host. This suggests that you also need to consider good change-control practices, to prevent many administrators from making changes on top of each other. If the administrators all share access to a single console of a host, they will know whether they have exclusive access to make their changes. Conserver can help whether the host is up or down.

Considering the Alternatives

You may have a bunch of hosts connected to monitors and keyboards. The monitors all take up space, use power, and create extra heat when they are turned on. If you aren't watching all the time, you probably rely on the scrollback in some of the GUI windows

on each host. The cost of each 15" screen and keyboard approaches the cost for a terminal-server port. In a large data center, add in the cost of air conditioning capacity for all those monitors. (You probably don't want more than a few display devices in a large data center.)

Keyboard/video/mouse (K/V/M) switch systems allow you to connect many hosts to a single display device. You'll find the cost of K/V/M ports are close to the cost of a terminal-server port if you are attaching PCs, and the cost is significantly higher to attach UNIX workstations (Sun, SGI, etc.) to a K/V/M switch. Don't forget the complication of screen resolutions, and scan rates! You can't get more than 12 to 18 devices on a big switch. While you can cascade the switches, it still doesn't scale for large data centers. You can find more information about K/V/M switches from the Celeste Stokely System Administration Web site.[4, 5]

VT-type terminals (or a terminal emulation window on a host) connected to a multiport switchbox give you text-only (CLI) support for much less money than a K/V/M, but you don't get GUI access when the host is up. The average cost for switchboxes with eight ports will cost you about the same as one terminal server port. Your limitations here include the physical distance for the serial console lines between hosts and the switchbox, as well as the number of devices you can attach to the switchbox. You can cascade these devices to gain more ports per terminal, but the length limitation is still there, plus you need to know which switch combinations will connect you to the port that you want.

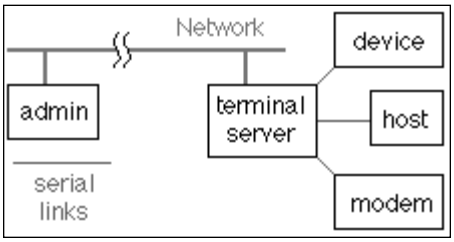
Terminal servers generate less heat than your average monitor, take up one network port, and allow you to access any serial port that you connect from anywhere on your network via a Telnet session. Most units take only one to rack units of vertical space to mount. Many sessions can be open to different ports at the same time, unlike most of the other alternatives above. (There are many Web sites that show how to connect various devices to various vendors' terminal servers.[5, 6]) The disadvantages to using terminal servers include the price, and the fact that there is no logging inherent in the terminal servers. If you were not connected to a console and watching, the data coming from each connected host is lost.

Adding the (free) Conserver application to a deployment of terminal servers adds the logging option, as well as adding a mentoring capacity, provides access control (and auditing of access) to the console ports, thus increasing the overall value of the deployment to the administrators.

The Value of Remote Console Access

If you are one of a few (or the only one) who carries a pager to respond to outages around your network, then remote access to your consoles should be one of the tools at your disposal. This gives you the ability to sit at any workstation (even dialed in from home) and be on the physical console of any connected host. This gives you faster visibility into your problem, and faster time to resolution, than if you needed to return to the data center to get to a terminal or keyboard. Do you remember a time when you were less than ten minutes from home, your pager went off, and you had to fight traffic back to the office to fix some problem? What would it be worth to you to keep driving home and get on the console remotely, with your dinner at your side?

Let's consider the benefits of deploying terminal servers for remote access to consoles, then we'll discuss the extra benefits of adding the Conserver software.



terminal server (no conserver)

Terminal Server-Only Communications Overview

- Device console ports are attached to terminal servers on the network.
- The administrator uses a Telnet session to connect to a specific serial port.
  - Telnet session is from admin's system to the terminal server.
  - Only one session at a time can be connected to each serial port.
  - If no session is connected to a port, data from the attached device is lost.

Imagine that you get home and access the console on the downed host. You hit return, and there is no response. If there is someone at the office whom you can call, you can ask them to cycle the power on your host, while you stay at home and watch the boot cycle. (Even if you don't have after-hours staff in the office there are power strips with serial ports[7], and/or Ethernet ports with Telnet and/or HTTP interfaces[8], to let you control power to individual outlets. You can cycle the power yourself, from home!)

Do you need to escort administrators into restricted areas, just for simple machine reset? We use console access to allow system administrators to control their hosts that need to live in a lab with restricted physical access. The ability to provide them with remote access to power-cycle the device as well as control the console has eliminated the need to allow nonessential staff to have physical access to security-sensitive areas.

While most UNIX hosts provide a method to use serial consoles instead of a video display and keyboard, your average PC platform does not have this built into the BIOS. While a UNIX kernel running on a PC will eventually allow you to use a serial port for a console, it isn't active until the kernel has booted, so you normally cannot see the Power-On Self Test information. But PC platform users do have some options!

Some PC server makers are modifying the BIOS to provide the ability to redirect the results of the Power-On Self Test (POST) to a serial port. Network Engines has recently added this, while Compaq has had this ability for about a year. Hewlett-Packard has an optional system management card that you can add to their servers. There are no standards for this type of console access, so the features for each vendor are different, and you should ask your vendor about the features they offer.

Tip: If your server vendor includes console redirection in BIOS, you should understand the limitations of their implementation. For example, the Network Engines BIOS hands off the startup to intelligent hard-disk controllers and Ethernet NICs, and the output from those interfaces is *not* redirected. If there are problems posted by those controllers, you will not see them through BIOS redirection in the current implementation.

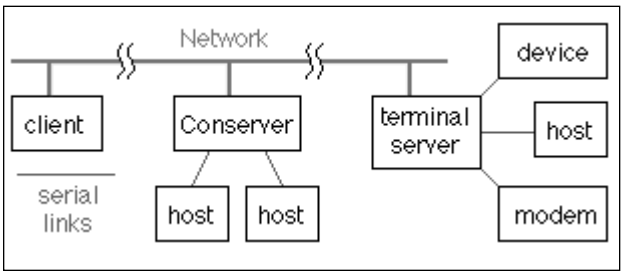
If you use a PC that doesn't have console support in the BIOS, but it does have a spare ISA slot, you can consider using the PC Weasel 2000 add-in card[9]. This appears to the server as a monochrome display adapter, but it translates the characters sent to the pseudo-screen into characters on a serial port. The PC Weasel also includes hardware that senses when your OS tries to use the serial port for its console, and it connects the serial port on the card to the operating system. The card also monitors the serial stream for a special sequence coming into the console, which allows the administrators to talk to the PC Weasel again to restart the PC (or the PC Weasel). Their Web site has an online demo.

Added Value from Free Software

Let's visit that downed host again. You connected to the console port, hit return, and nothing came back. What caused it to stop? If nobody was connected and watching, anything the host had sent to the console was lost. The logging capability of a console server application is one of the best reasons to combine it with your terminal servers.

Terminal Servers with Conserver Overview

- Terminal servers are deployed near the devices you want to monitor.
- The Conserver host is configured for the devices you want to monitor.
  - Hosts can attach to serial ports on the Conserver host.
  - Hosts can attach to serial ports on the terminal servers.
  - Conserver opens a socket-based session to each terminal server port.
- The client application connects to the Conserver via Telnet today (optionally by SSL soon).
  - The client session requests to be connected to a host logging session.
  - Many clients can connect to the same device session simultaneously.
  - Clients can connect to multiple distributed Conserver hosts.



conserver overview

With Conserver in this equation, you connect to the downed host, press return, and get nothing back. Then you use a few meta-keystokes to replay the last 20 or 60 lines of the log for that device to your session. This will usually tell you why it stopped (bad memory, full disk, other problems), and this may affect your decision to just cycle the power. You may decide that something needs to be fixed before you bring it up again.

You can deploy terminal servers, and distributed console servers, across the country, and even internationally. This can be an important tool if you have to support smaller, remote offices without administrators onsite. Sure, you can preconfigure a new host for a remote office, and even set it up in a test lab. You can include diagrams and documentation for the remote-office staff, directing them how to unpack it and plug it in. But when they follow your instructions, and you can't connect to it, what do you do? Before you can talk someone through fixing the problem, you need to troubleshoot it.

- Is it powered?
- Is the network connection plugged into the correct interface?
- Is there a duplicate network address in the office?
- Did the network switch auto-negotiate to the wrong speed?
- Was the disk damaged in shipment?
- What will the remote office folks use for a console, in order to be your eyes?

If you had it plugged into a terminal-server port in that office, you could get most of those answers yourself, rather than talking the office staff through the troubleshooting steps. (Now how much is that terminal server worth to you?)

You can find links to a number of console server applications on the Console Connection Guide Web pages[6], but my favorite application is currently Conserver for a number of reasons:

1. Conserver allows you to control devices attached to serial ports connected to the Conserver host, as well as to serial ports on terminal servers scattered throughout your network.

2. Everything that comes out of a device's console port goes into the Conserver log file for that device.

3. Single-user control, but multi-user viewing, makes Conserver an excellent mentoring tool. Control can be easily switched between users, allowing collaboration between administrators in different locations.

We have used Conserver during maintenance downtimes, along with a conference call, to allow a standby administrator to hear and watch the progress of an upgrade

session. This was done because the person making the changes was across the country, while the standby person was at his desk in the same building as the equipment, in case the hardware failed during the changes.

4. The log files for various devices can also be used as a teaching tool, as a junior administrator can look through sessions to see how a senior administrator performed some tasks, providing that your security policies and the file permissions allow this. (Conserver logs the data coming in from the attached host, not the data going from clients to the host.)

You can also use Conserver logs for training after a failure. If a junior-level administrator started the troubleshooting, and a senior had to come in later and follow up, both administrators can benefit. The senior can look at the logs and see what was already done (including looking to see what happened before the failure). The junior can watch the senior and perhaps learn new methods to use the next time.

5. Scripting tools can also sift through the logs, looking for problems, as a backup to your other automated device managers. These includes SWATCH10, presented at LISA in 1993.

6. The logs can also provide additional logging, as a backup to your syslog files. (A cracker can find out where you are sending syslog messages, but the host doesn't require any special pointers to the Conserver host in order to be connected and logged, so a cracker would not know there was another log to clean up after an intrusion.)

We have found syslog settings misconfigured on some hosts more than once using the Conserver logs. While one machine kept quietly rebooting at random intervals, the administrators found nothing in the syslog files, yet the conserver logs told two stories: first, the cause of the rebooting was a failure to write to a full file system; second, the fact that syslog wasn't logging the errors led us to check the syslog configurations.

7. If a device attached to your Conserver doesn't timestamp its output, Conserver can be set up to put an hourly timestamp in the log for that device.

8. Multiple distributed Conserver hosts can be controlled by a single configuration file, which specifies which servers are connected to the various hosts and devices. This provides easy configuration, even when you deploy Conservers in remote offices.

Tip: Deploying a Conserver host at each remote office means that console data won't be lost if a WAN link fails. If you allow remote access to the remote office, you can look at the logs during the link failure to see what activity may have happened and keep an eye on your link providers work during the repair.

If the WAN link to a remote office goes down, you can still dial into a modem attached to the terminal server in that office, pass the authentication challenge, allowing the network administrator to look at the CSU/DSU and router on both ends of the failing link at once.

9. You can't use grep(1) on a pile of paper on the back end of a DECWriter! But you can use it on the log files from Conserver.

Tip: As with syslog and backups, keeping the clocks between all of your Conserver-connected systems in sync is very important. We recommend a stable NTP infra-

structure be in place when you deploy a console server. During large problem events (denial of service attacks, network outages, cascading failures across multiple servers), your troubleshooting will be faster if all of the clocks are in sync. This allows you to correlate time stamps between various hosts and network devices, and to understand what happened first and what happened after that. (Consider using one time zone for all hosts if making the time-zone conversions for troubleshooting is a concern for you.)

The maintainer of Conserver and I will present a half-day tutorial about deploying remote serial console access at the LISA conference[11] in December. We invite anyone with an interest in the topic to sign up. We will be covering several models for deploying Conserver, and we'll try to take the mystery out of connecting various serial devices to your terminal servers.

There are also options that you can use for added security, but architecture also plays a part in that discussion. Deployment models vary, depending on the security needs. Bring your questions to LISA, and look for the Conserver BoF session!

What about the BREAK Problem?

If you have Sun hosts, you may have been warned away from attaching terminal servers for remote access. Let me offer a brief explanation and some information that may help.

There is a serial-port equivalent to the Telnet BREAK signal, where the data lead signal is inverted from its normal state for a brief period of time (more than the duration of a single character, including start and stop bits)[12]. This is not the same as sending control characters. Most terminal servers send a serial BREAK to every port when you turn the power off, and some even do it when you turn power on, or during their boot sequence. This problem also exists on most multi-port add-in cards for PCs.

When Sun machines receive a serial BREAK, they will drop down to the "ok>" prompt. This stops the operating system, and all the useful services that the machine is doing at the time, until someone gets on the console and types "go." This is actually quite useful for getting the machine out of a hung state, so you normally don't want to block this signal. Newer versions of SunOS either allow a patch[13], or include in the OS the ability to ignore the serial BREAK but listen for a specific character sequence instead. Sun customers with SunSolve[14] support access can check SunSolve for more information.

The problem occurs when you have a bunch of Sun hosts connected to a terminal server, and then the terminal server sends a BREAK to all of the attached hosts – and everything stops until you get on each of the consoles and type "go" for each host. (You can infer why it would be a bad idea to plug the console of your Conserver host into a terminal server that you access through that Conserver.)

Cisco Systems has also posted a Field Notice related to the BREAK problem to a Web page[15] in April 1998. The basic idea behind the notice was trying to keep the BREAK signal from getting to the Sun host, or make the Sun host ignore the signal. Unfortunately, the signal is too useful to administrators, so we can't advocate methods that block the BREAK signal from getting to the host.

Our search for a good answer has led us to try to find terminal servers that don't send BREAK at inopportune times. In support of Mark Burgess's series of articles (Systems Administration Research) in recent issues of *;login:*, I have started a series of tests on a variety of terminal-server models. We're posting our test methods as well as the results,

and encouraging other sites with an interest to perform similar tests and share their results with us. We're trying to determine whether the failures for various vendors are related to hardware or software; trying to determine when you can expect the failure to occur; and trying various recommended settings on the terminal servers to try to eliminate the problem. You can find more information on our results page[16].

In Summary

If you have always worked at a site with remote access to the serial consoles, consider yourself lucky. If you don't have some type of access now, I hope I've given you some reasons to consider adding it soon. If I've managed to interest you in the subject, but you want to learn more, I hope to see you in the tutorial in December.

I believe it was Elizabeth Zwicky who wrote a series of articles about software that should have been included in our favorite operating systems. I think Conserver belongs in that category.

References

1. Thomas A. Fine and Steven M.Romig, The Ohio State University: "A Console Server." Found in hardcopy proceedings of LISA IV, October 17-19, 1990. It is not currently online at the USENIX site.

2. Historical notes: Ohio State distributed the original terminal-server code. Conserver was the name of the server component. Folks at Purdue University added a "pipe to shell" feature, which some folks used to send sessions through Telnet sessions to high TCP ports in order to reach console servers. Later, Robert Olsen at Argonne National Labs (ANL) hacked the OIS code to include the socket-based support. The Certainty Solutions (formerly GNAC) version derives from a Purdue release, with the ANL additions, plus contributions from Arnold de Leon while at Synopsys, Inc.

Package	Maintainer	URL
Conserver	Thomas A Fine	<http://hea-www.harvard.edu/~fine/Tech/console-server.html>
Purdue version	Kevin S. Braunsdorf	<ftp://ftp.physics.purdue.edu/pub/pundits/>
Certainty Solutions (formerly GNAC) version	Bryan Stansell	<http://www.conserver.com/>

3. Conserver application <http://www.conserver.com/>

4. Celeste Stokely Sysadmin Web site <http://www.stokely.com/>

5. Celeste Stokely UNIX Serial Port page <http://www.stokely.com/unix.serial.port.resources/tutorials.html>

6. David K. Z. Harris's Console Connection Guide Web pages <http://www.certaintysolutions.com/consoles/>

7. BayTech Data Communications Division, Bay Saint Louis, Missouri, USA. RPC series power control devices. <http://www.baytechdcd.com/products/rpcseries.shtml>

8. American Power Conversion Corp., West Kingston, Rhode Island, USA. Master Switch, and Master Switch Plus units. <http://www.apcc.com/products/masterswitch/index.cfm>

9. Calgary Connect Corporation, Calgary, Alberta, Canada. PC Weasel 2000. <http://www.realweasel.com/>

10. SWATCH <http://www.stanford.edu/~atkins/swatch/>

11. USENIX LISA 2000 conference. <http://www.usenix.org/events/lisa2000/>)

12. "Communications Concepts, An Introduction to Data Communications", Communications Research Group, and Telebit Corp., pp. 3–12

13. Sun part number for the Consulting Special to address the serial BREAK problem: "CONSULT-ZSBRK"

14. SunSolve contract support site <http://sunsolve.sun.com/>

15. Cisco Field Notice about fixing the BREAK problem on Suns <http://www.cisco.com/warp/public/770/fn-tsbreak.html>

16. Certainty Solutions (formerly GNAC) Terminal Server BREAK-off pages <http://www. certaintysolutions.com/consoles/breakoff.html>

sysadmin ethics

Recently I have been thinking about sysadmin ethics. Here are some of the things I've been thinking about, and some conclusions I've come to.

Opinion by David Parter

David Parter is a member of the SAGE Executive Committee.

<*dparter@cs.wisc.edu*>

Ethics has been part of SAGE since the beginning. SAGE-AU adopted an ethics code several years ago. SAGE accepted an ethics document for discussion and has published a code of ethics. Recently, a group has been working on a unified ethics document for all the international SAGE groups.

1. I have been served well by the SAGE ethics document. I used it to jumpstart a discussion on sysadmin ethics with the student sysadmin staff at the university where I work. I don't recall what prompted this particular discussion – it wasn't an incident involving our staff. I thought that the discussion would be more productive than a lecture about ethics from me.

2. I have heard from several friends that the SAGE code of ethics has been useful in explaining to others (usually management) why they should not or will not do something that they have been asked or told to do.

3. I think the SAGE ethics document has served SAGE well. At numerous events, I have explained SAGE to potential members and others in related areas of the computer industry. I always mention the ethics documents. It has always been a positive with the people I'm talking to. I think just mentioning our ethics document has helped a lot of people to think about sysadmin as a serious profession.

4. Last May, I gave a talk to the Twin Cities System Administrators (TCSA), the Minneapolis/St. Paul SAGE local group. My talk was a SAGE activities update generally and a discussion of the SAGE certification program. One of the participants (not a SAGE member, but I think he's joined since then) asked about ethics. Would it be a component of the certification process. Wow. I hadn't thought about that. (Note – I'm not on the certification committee. They may have had a discussion about it, I'm not sure).

5. At the USENIX Security Conference I was part of a fascinating discussion in the bar. The issue was the ethics and morality of certain security practices. Full disclosure vs. limited disclosure. "Grey Hat" security people. The discussion was wide-ranging, and touched on many situations, and drew upon many analogies. Case in point: a common network scanning tool that is extremely useful, but also has some stealth characteristics. For the sysadmin or "white hat" security professional, the tool would be (or should be – more on that later) equally valuable without the stealth features. Why are they there, if not to make the tool equally useful to the black hats?

6. A friend was sitting next to me during this discussion. He commented that the stealth features were useful to him – it prevented another organization within his company from knowing that he was port-scanning his own network. It is important to note that this was not a case of hiding it from his users, but of hiding it from another IT group that might object or think he was out of line for taking responsibility for the network for which he is nominally responsible.

7. In various conversations, we – the sysadmin community at large – have been accused of compromising on ethics. Why? Because every time someone poses an ethical dilemma on the <*sage-members*> mailing list (or in a BoF or tutorial) almost all the answers include disclaimers ("I don't know the details of your situation," "You will have to decide how important this issue is to you," "You might want to consider

looking for another job"), and refuse to consider the question or its answer in "black-and-white" and flat out tell the questioner what to do.

What do I conclude from this?

Ethics is an important part of the profession. Ethics is also important to me as a person. Can I separate the two? Maybe, but not entirely. My own personal ethics will guide me in any discussion of ethics for the profession, or for the workplace and job situations I find myself in.

Ethics are important, but complicated. I don't think we are compromising our ethics (as accused in #7 above) when we recognize that not all situations are the same, and not every sysadmin shares one's particular personal ethical system. We may have a consensus on a general statement of ethics, but the devil is always in the details. Is divulging the contents of a user's email an ethical violation? It depends on the circumstances. To whom are you divulging that information? Why? What does the company/site policy say on the subject? These are just a few of the questions that have to be considered before a judgment can be made. It has been observed that the "tech culture" has a strong streak of libertarianism in it. Many of us are fierce advocates of civil liberties, objecting to electronic (and other) censorship and invasion of privacy. Yet that same streak prevents us from imposing our values on others. I don't think that is a cop-out.

In response to this issue, it was suggested (in one of those late-evening discussions among fellow sysadmins) that we borrow from the legal and medical professions. They (according to the person who proposed this – I haven't researched the details) have "ethics boards" that can discuss the details of a situation with a member who is facing an ethical dilemma, and based on the profession's code of ethics, traditions, and personal experience (and probably the law), give guidance. The entire discussion must, of course, remain confidential.

Would such a system work for sysadmins? I don't think we are ready to do that on a formal basis. But many of us, through our networks of colleagues (from LISA conferences, coworkers at past jobs, local groups, etc.), do have a resource that we can call upon to help us face these decisions. The result may not have the weight of an "ethics board" ruling, but it seems to me to be an appropriate step to take. This assumes that there is a common understanding of the basic ethics of the profession, and a recognition of the role of ethics in the profession – at least among those one chooses to consult.

Not every sysadmin has a well-developed network of peers to consult in such circumstances – even those who do consult the <*sage-members*> mailing list. Hopefully, the answers and discussion on the list have helped members deal with their situations. I assume that in some cases there is private follow-up between the poster and some of the respondents, to provide the type of advice that requires more details.

Can we (society at large) "teach ethics"? I don't know. I have never had an ethics course, but I know they exist. I don't know what is in them. I have always assumed that we can't "teach ethics" (and have as an outcome ethical behavior by the students) by the lecture or pronouncement method. I think each person develops his or her own personal code of ethics, and will most likely resist a rigid code being imposed upon them. I think we can teach about ethics and raise the issues, and raise the awareness among our students (or membership, or users, or management). We can – indeed must – also teach by personal example. As an example, I am 100% sure that every sysadmin on the

staff where I work knows that I am extremely concerned about protecting our users' privacy, and that it is based on more than just the university's rules.

Should we include ethics in the certification process? Yes. Should we judge the applicants' ethics? No. We are not at the point where the ethics of systems administration are universally understood and agreed to by sysadmins, their users, managers, and the general public. Until we get to that point – which will probably take a long time – the profession and professional bodies are not in a position to judge. We can advise, educate, and discuss. In our certification process we can ask applicants about their knowledge of the SAGE code of ethics, but that is all.

If you have not looked at the SAGE code of ethics recently, you should: <*http://www.usenix.org/sage/publications/code_of_ethics.html*>). Have you had an ethics discussion (not a lecture!) at work? In your local SAGE group? It might be time to do that.

just presuppose...

by Steve Johnson

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

<scj@transmeta.com>

and Dusty White

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

<dustywhite@earthlink.net>

Have you ever been asked by someone, in a joking tone, "Are you still driving that old lemon?" And experienced a moment of blankness as you realized that neither "yes" or "no" felt very good to you? There are many such questions (many people are familiar with the old question "Have you stopped beating your wife?", which is of the same type). But few people take the time to analyze such sentences to understand why they are so difficult to deal with. That is our topic for this month.

All meaning is context dependent. A sentence such as "They are visiting relatives" can have several different meanings, depending not only on the linguistic context but also on nonverbal context such as a pointing finger. Frequently, however, in order to make any sense out of a sentence, we must accept that certain things are true in the context of that sentence. A sentence such as "I gave John a rose from my rosebush" requires you to accept that I have a rosebush and that the bush had a rose on it. Otherwise, you can't make sense of the sentence. These assumptions are called presuppositions, and they are a very powerful way of pulling the wool over someone's eyes. Note that it is much easier linguistically to dispute that I gave John a rose ("No you didn't!") than that I have a rosebush (ahh, er, about that rosebush . . .).

So the question "Are you still driving that lemon?" has a significant presupposition – that the car in question is a lemon. To understand the sentence, you must accept the presupposition. And then it doesn't matter whether you answer yes or no – your car is still a lemon.

Understanding presuppositions is important for two reasons. One is that they are useful, especially in the persuasive arts. The second is that they may be used against you, and you need to know how to defend against them. You go to buy a car, and the car dealer winds up his pitch by asking "Would you like the red one or the blue one?" The presupposition is that you are already going to buy. Many people are swept along and into the manager's office without ever clearly having a moment when they decided to buy. The dealers love it that way.

Presuppositions can be exquisitely subtle. Consider this gem, collected by Tad James: "What's the one question that, when you ask it, will totally address all your objections and allow you to buy this car?" A careful reading will reveal that, if you accept the presuppositions, you will buy the car even if your question is never answered. In fact, even saying "I don't know, what?" presupposes that you will buy the car. Whew!

Presuppositions are also a frequent source of misunderstanding and confusion in both business and personal communication. A man who asks his wife "What are we having for dinner tonight?" may fail to realize the presuppositions – that it is his wife's job to decide what is for dinner, that it's OK for him not to know or care until the last minute. His wife may also not be aware of the presuppositions either, consciously, and just find herself growing irritated. The courts handle thousands of contract disputes a year because some presupposition was not written into a contract, and the two parties disagreed about how to deal with the resulting situation. Even in everyday business, being attentive to presuppositions is important. An employee was hired recently at a high tech company with the hiring managers and all the interviewers presupposing that he knew C, only to discover he was completely ignorant of it (he was a crack Lisp programmer, but they never asked and he never told).

So what can you do when you suspect that you are the victim of a presupposition apparition?

The first thing to do is to stop whatever you are doing, and set about slowly and methodically to analyze the statements, bringing the presuppositions out into the open. Check these with the other person – "It sounds like you are assuming that I will buy this car. Actually, I still have reservations about the price." Sometimes people are very calculating with their presuppositions; other times people simply hear what they want to hear – the dealer may have genuinely believed that you said you would buy (yeah, right, we can hear you say. But it doesn't hurt to assume the best and watch your back).

Often our old friend Chunking (see *;login:* Vol. 25, No. 1, p. 64) can be used to get out of a double bind gracefully. By talking about the color of the car, the dealer is chunking down, getting more detailed. You can chunk up by saying things like "What other fuel efficient vehicles do you have on the lot?" or "Are your prices generally lower in the summer?" This also gives you a chance to dodge the presupposition without acknowledging its existence, a valuable skill when tact is called for.

You can also use presuppositions in your own communications. When the situation is a win/win, they can be a way of saving time, increasing motivation, and sending messages that might be awkward to send more explicitly. For example, a manager may ask an employee "What job in this company would you like to be in five years from now?" There are some major presuppositions in this sentence – that the company will still be in business in five years, that the employee will still be working for it in five years, and even a presupposition that the manager will still be with the company in five years and can do anything effective with the answer to the question.

By answering this question, the employee accepts these presuppositions, implying that their future lies with the company. Subtle? Yes. Ethical? Probably, since most employees are able to frame the discussion with an understanding that the future is unpredictable. And the biggest presupposition, that the manager is interested in fostering the employee's career growth, may be the most important message of all.

the bookworm

by Peter H. Salus

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

<peter@pedant.com>

Well, it may not be the end of the year yet, but I know that the new edition of *Building Internet Firewalls* will make my top-ten list!

In the five years since the first edition of Chapman & Zwicky, *Firewalls* has picked up an author (Simon Cooper) and added about 350 pages. It's a good deal heftier than it was, but it's really packed with useful information.

If you're at all concerned with security, firewalls are a necessity. And this book is a must read.

I don't intend to go through every chapter, or even most chapters, here, but I do want to point out some exceptional aspects of Zwicky-Cooper-Chapman. Not at all least among these are the appendices: three of them, comprising over 50 pages. First, a truly superb list of resources – Web pages, FTP sites, mailing lists, papers, articles, books, etc. Second, tools and packages with complete URLs. Third, a (very) brief general essay on cryptography (with a pointer to Bruce Schneier's excellent *Applied Cryptography*).

I found chapters 10, 11, and 12 (on Bastion Hosts) very interesting. It's a topic I'd never thought about. Zwicky, Cooper, and Chapman have made the exposition lucid, and by covering (as they do in many places) UNIX/Linux and Windows (NT and 2000).

Also notable is chapter 14 on intermediary protocols. As most of you know, I'm addicted to RFCs and protocols. Reading the expositions here was genuinely illuminating.

You might even copy the 25 pages of Chapter 1 for your CIO or VP: it's clear and concise enough for him/her to understand.

This is an important and worthwhile book. Thanks, Elizabeth, Simon, and Brent.

Punditry

Another important book, for a different reason, is Metcalfe's anthology of *InfoWorld* columns. But Metcalfe, while egocentric, infuriating, brash, and irritating, is also interesting, entertaining, and (frequently) authoritative. As the essays are short, this is the perfect book to take on vacation or on a plane trip. You will surely snort occasionally; you'll guffaw sometimes, and you'll find that you even agree now and then. The most important thing is, however, that you recognize that a lot of the time Metcalfe is provocative yet right.

Microsoft was abusing its monopoly in 1991. The Internet did not collapse in 1996. Those are two examples.

Bob Metcalfe is an egotistic gasbag. Read him!

Useful Stuff

If you live on the Internet, you know that there are problems. Yes, we've got trouble right here in River City! The result of this is that books that offer solutions to real-world problems are of great value.

Two such books that have come my way in the last few months are Petersen's *Unix Networking Clearly Explained* and Ward's *Linux Problem Solver*.

Petersen really knows his stuff, and it shows. It shows most significantly in his breadth. He talks about mail, and then there are chapters on mailx, Elm, MH,

BOOKS REVIEWED IN THIS COLUMN

BUILDING INTERNET FIREWALLS, 2ND ED.

ELIZABETH ZWICKY, SIMON COOPER & D. BRENT CHAPMAN

Sebastopol, CA: O'Reilly, 2000. Pp. 869. ISBN 1-56592-871-7.

INTERNET COLLAPSES AND OTHER INFOWORLD PUNDITRY

BOB METCALFE

Foster City, CA: IDG Books, 2000. Pp. 324. ISBN 0-7645-3503-X.

ENIAC

SCOTT MCCARTNEY

New York: Walker & Co., 1999. Pp. 262. ISBN 0-8027-1348-3.

UNIX NETWORKING CLEARLY EXPLAINED

RICHARD L. PETERSEN

San Francisco, CA: Morgan Kaufmann, 1999. Pp. 591. ISBN 0-12-552145-6.

THE LINUX PROBLEM SOLVER

BRIAN WARD

San Francisco, CA: No Starch Press, 2000. Pp. 283 + CR-ROM. ISBN 1-886411-35-2.

ESSENTIAL XML

DON BOX, AARON SKONNARD & JOHN LAM

Boston, MA: Addison-Wesley, 2000. Pp. 368. ISBN 1-201-70914-7.

and Pine. Where news is concerned, rn, readnews, trn, tin, and nn are all there. And archie, WAIS, Gopher, Veronica are discussed along with WWW.

There's a lot more, but I was really pleased to see that stuff that's more than a few years old, but still in use, has not been totally forgotten. I still like Lynx, for example; I'm frequently uninterested in the zillions of bits of graphics.

Ward is useful for a very different reason: there are over 100 books on installing some flavor of Linux and getting started, but there's too little on what you do next – what you do when there's a real problem with your network or you've suffered a system crash.

Ward answers the questions you'll have in real life, after your Linux system is up and running. The CD is useful, too, as it contains a bunch of config files and can serve as an emergency boot disk.

Petersen and Ward share the shortcoming of not have a references section, neither of URLs nor of books.

XML

Box et al. have written a book that's not so much about the Web, as about how XML can be utilized as "universal duct tape for all software integration problems." I'm not really convinced as to XML's universality, but I have been sold on its usefulness for several years.

This is a fine book on a relatively high level, covering a number of fundamental abstractions and the concepts that underlie XML technology.

The three appendices cover the XML information set (complete with URLs), XML productions (both xml and xmlns), and an "example gallery" – all in under 100 pages.

Highly recommended.

SAGE news

SAGE Code of Ethics

by Barbara Dijker

SAGE President

<barb@sage.org>

and Lee Damon

Lee Damon is chair of the SAGE Ethics Working Group, and was one of the commentators on the original SAGE Ethics document.

<nomad@castle.org>

For a professional body, an ethical code is an important part of defining that profession. For more than four years, a Code of Ethics has been posted on the SAGE Web site. It was the result of years of collaboration among SAGE members interested in that effort. At each LISA conference since 1994, an Ethics BoF has been held to inform members on the status of the project, get comments, and attract new blood to the ongoing Ethics Working Group and its associated <*sage-ethics*> mailing list.

System administration is a task that is ubiquitous in a computerized world. SAGE has encouraged and supported the development of SAGE groups in other parts of the globe. There are currently three formalized SAGE organizations outside the US. There are others in the making and yet others either loosely organized or comingled with other bodies. The global expansion of SAGE is expected to continue.

Ethics, one would hope, transcends law and culture. So it would follow that there should be one code for all system administrators, or at least all regional SAGE groups. The existing SAGE Code of Ethics was developed independently of the one developed for SAGE-AU. The result was two documents. Even before the code was effectively completed in early 1997, those involved with the project recognized the need to rewrite it from scratch.

The SAGE and SAGE-AU documents are effectively the same or as different as night and day, depending on whom you ask. Either way, it quickly became clear that neither side was likely to abandon its code and adopt the other without significant revision.

At the direction of the SAGE Executive Committee, the Ethics Working Group then began work on a new global code. The start of this rewrite was first discussed in detail at the Ethics BoF held at LISA '97. The next Ethics BoF at LISA '98 was hosted by Hal Miller, who shepherded the first ethics document, and Geoff Halprin, of SAGE-AU. At that BoF, Lee Damon was asked to coordinate the effort of drafting a new global Code of Ethics.

A new ten-stanza draft ethics document was presented at the Ethics BoF at LISA '99 in Seattle. The first six stanzas were reviewed, discussed, debated, voted on, and approved at that BoF. When time ran out, the discussion of the remainder of the new draft was moved online. All of the people at the '99 BoF were asked to sign up for the mailing list, and along with the attendees from the '98 BoF, became the new Ethics Working Group. The working group continued going over the remaining stanzas one at a time, rewriting sections and voting on each stanza in turn.

At press time, the working group is voting on the last stanza of the new draft code. The final report of the Ethics Working Group, including the proposed new code, should be available in the December issue of *;login:*. In the coming months, the drafted new code will be reviewed by representatives from each of the regional SAGE groups. A process for adopting the code will be developed by each regional group. Information about this project will be posted at <*http://www.sage.org/ethics/*>.

We would like to thank the 34 stalwart individuals who took so much of their time to help craft the initial draft of the new global document.

SAGE Certification Process Proceeding

by Phil Scarr

Phil Scarr is a senior systems architect at Certainty Solutions (formerly GNAC, Inc.). He's been involved with USENIX and SAGE for several years and is co-chair of LISA 2000.

<prscarr@greymouser.com>

The sun was shining brightly in Seattle (for a change) on August 1st for the SAGE Certification Policy Committee meeting.

Many of you may be wondering why I was there since I have been a strong objector to the certification process. I was one of the nay sayers at the "Great Certification Debate" at LISA and I still maintain a strong sense of skepticism towards the value of certification in our profession. However, since the certification ship was getting ready to sail with or without me, I decided the prudent thing to do would be to try and help steer it through the shoals in the hope that SAGE will deliver a strong certification program tightly coupled with a strong educational program and not just another rubber stamp certification.

The attendees at the Seattle meeting were:

Lois Bennett, Stephen Berry, Mark Burgess, J. K. Chapman, Barb Dijker, Bradley Donison, Tim Gassaway, Richard Jaross, Mark Langston, Phil Scarr, Mark Stingley, John Stoffel, Leeland G. Artra, Gale Berkowitz, USENIX; Ellie Young, USENIX; Geoff Halprin, SAGE Executive

Committee; Michael Hamm, Consultant; Gordon Waugh, HumRRO.

The committee nominated J. K. Chapman as the Committee Chair.

The primary goals of this meeting were to deliver a plan of action for the development of a business plan, develop policies and procedures, and define criteria for the Exam Development Committee.

The committee reviewed and debated many aspects of the certification question. But a general consensus was achieved that there is sufficient interest among the members of SAGE and USENIX to provide a certification program for the members. This was borne out by the feasability study commissioned by SAGE.

In order to deliver this program, there are numerous details to be worked out by the Policy Committee. For instance, there must be an administrative framework to manage and deliver the examinations. This framework includes both professional staff as well as volunteers from the ranks of SAGE and USENIX. The exact composition of this framework is still being analyzed.

Through the process of defining the characteristics of a certification program, we were asked by Michael Hamm, a consultant specializing in professional certification programs, to review several important planning questions. Among them were: The motivations for a certification program, objections to such a program, levels of certification, competition, cost, and the key question of recertification. Here are summaries of the answers to these questions.

What are the motivations for a certification program? There are several, among which are: The ability to objectively measure skills; a response to both member and market needs; to advance the profession; to "set the standard" for system administrator certification; to foster a philosophy of personal and professional development in the field; and to help focus the educational programs within SAGE/USENIX. This last point is one that is key to delivering a sound certification program.

What are the objections to a certification program? Again, there are several, among them are: It simply can't be done; the field is changing too rapidly for a certification program to keep up; it legislates mediocrity; it can lead to exclusionary behavior.

What are the levels of certification? The program will follow along the lines of the existing SAGE Levels for System Administrators.

Who is the competition? The committee identified several key competitors in this field. Among them are the ACM, Universities, and SANS. But in the field of general system administration, there are few non-vendor programs. However, one of the big educational companies, Learning Tree International, is very interested in using the SAGE/USENIX certification exams in their own courses and to have SAGE evaluate their coursework for completeness. There is nothing formal, but it could be an interesting project.

What is the cost? There was a lot of debate on this question and nothing conclusive came out. This issue will be raised again in October at the next meeting.

What about recertification? It was generally agreed that the credentials would have an expiration of no more than 3 years and that "points" (like Continuing Education Units) could be used to provide that recertification. These points would be awarded for the completion of training courses, teaching classes and giving talks. However the mechanism for achieving this remains undefined.

Finally, the committee reviewed the plans to create the exams themselves. Gordon Waugh from HumRRO described the process of creating and managing the exams. They will all be multiple-choice exams (to begin with) and we will be developing the first one at the very basic level. There will be two forms of the test and 150 items per exam.

There was a lot of debate on the question of multiple-choice versus more hands-on approaches. There were several people who felt that without a hands-on component, the certification process would be incomplete. However, the cost (to both USENIX and to the participants in the program) of such a hands-on exam is prohibitive. However, as the certification program expands to cover more senior system administrators, such a scheme will be revisited. Most people agreed that to deliver a certification with the clout of the Cisco CCIE, hands-on examinations were required.

The formation of a Test Development Committee is underway. This committee will be responsible for delivering the questions for the question pool. Membership requirements for the Test Development Committee were reviewed and will include Subject Matter Experts, people who are analytical, have diverse experience, experience evaluating system administrators, self-critical, clever (to help write wrong answers) and at least 5 years of sysadmin experience. John Stoffel will chair the search committee with Tim Gassaway and Louis Bennett helping to coordinate the search and Gale Berkowitz as the staff coordinator.

A tentative timeline for the entire process was worked out:

Mid-August 2000: Receipt of Business Plan from consultant

Late August 2000: Certification Committee to review draft of business plan

August 22, 2000: Submit names for potential Exam Development Committee members

September 8, 2000: Selection of Exam Development Committee

October/November, 2000: Convene first meeting of Exam Development Committee to conduct training

October 21, 2000 (San Diego) (tentative): Convene next meeting of SAGE Certification Committee

November, 2000: Item writing takes place

December 2000 (at LISA) (tentative): Convene second meeting of Exam Development Committee to review questions

May 1, 2001: Selection of pilot test questions

June 2001: Pilot testing takes place

Fall 2001: Rollout of first exam

While this is just a tentative timeline and things may change, delivery of the first exam should take place in Q3 or Q4 of 2001.

SAGE Volunteers Needed!

Want to get involved in SAGE, but don't know where to start? Here are a few places where SAGE can use folks with some time and energy.

SAGE Annual Awards

<www.usenix.org/sage/people/awards.html>
Volunteers are needed to help decide the recipient(s) of the SAGE Outstanding Achievement Award, to be given at LISA in New Orleans.

SAGE Elections

Early in 2001, elections will be held for the SAGE Executive Committee. If you want to be on the Nominating Committee, or are interested in running for office, please contact the SAGE Nominating Committee.
<nomcom@sage.org>

Mentors

The mentoring project
<http://www.usenix.org/sage/mentor/index.html>
needs people who are willing to be mentors. This is a great chance to give back and have a hand in the future of the profession.

The USENIX Conference Office Has Moved!!!

Please note the new address and phone number for USENIX conference management services:

USENIX Conference Department
2560 Ninth Street, Suite 215
Berkeley, CA 94710

Phone: 1.510.528.8649

Fax: 1.510.548.5738

Email: *<conference@usenix.org>*

USENIX news

20 Years Ago in UNIX

by Peter H. Salus

USENIX Historian

<peter@Matrix.Net>

One of the really nice things about publishing this set of historical notes is the way the participants contribute to my knowledge.

For years I've been reading columns by the "Jeffreys, Haemer and Copeland." Jeff Haemer is an old friend, but I'd never met Jeff Copeland. However, in June I received the following:

> In your June "20 years ago . . ." column, you quote some correspondence from Ted Dolotta on the use of the word "touch" in Interactive's documentation for IBM. Ted minimizes his own role in preventing the travesty.
>
> Certainly, Joyce Yoshihata uttering that she had been a press-typist for 25 years stopped the first round of IBM insisting on changing "touch" to "press," and that solved the problem with the IBM documentation managers. Unfortunately, as in many a large organization, the word didn't get passed down to the troops. As a result, when the documentation review for the text processing tools (with which I had some ego involvement) rolled around, the first words out of the young technical editor sitting across the table were something like "The first global issue is your pervasive use of the word 'touch' – that's going to have to be changed."
>
> Ted by then had had enough. He leaned his elbows on the table, looked the editor in the eye, and said "OK. Fair enough. How about 'fondle?' Can we 'fondle the key?'"
>
> After she turned seven shades of pink, Ted explained that (1) we'd been through this before, and (2) if AT&T – the other conservative high-tech monopoly in the country at the time – could sell "Reach Out and Touch Someone" nightshirts in their company store, perhaps IBM should loosen up, too.

Soon after I received this, I had the opportunity to actually meet Jeff, who is currently with a large (unmentionable) corporation in Redmond, Washington, whose opinions his email does not represent. I can assure you all that he is as witty in person as in print.

While this most likely ends the Dolotta documentation tale, it far from concludes this episode of history.

Fall 1980 saw Sam Leffler's arrival on the Berkeley campus. In October, the CSRG released 4BSD. This release contained a faster filesystem for use with virtual memory, job control, reliable signals, automatic reboot, "delivermail," and the Franz Lisp system.

Eric Allman's delivermail is what was renamed sendmail. (Eric said: "sendmail is really just delivermail version 2 or 3.") (The full story is in *Quarter-Century of UNIX*, pp. 161-163.)

Finally, Fall 1980 saw the resumption of the publication of *;login:* – I'll leave that for the next installment.

USENIX Association Financial Report 1999

The following information is provided as an annual report of the USENIX Association and represents the Association's statement of revenue and expenses for the year. Below are several charts that illustrate where your membership dues go. The Association's complete financial statements for the fiscal year ended December 31, 1999 are available on request from the USENIX Association at 2560 Ninth Street, Suite 215, Berkeley, CA, 94710.

The USENIX Association completed fiscal year 1999 with a net operating surplus of $117,361.

Membership increased again, both in the general membership and SAGE, and now tops 10,000 members. About half of the members are also members of SAGE. However, with increasing membership, comes growth in expenses. Membership and publication expenses increased.

Member Dues

Here are a few charts that show how your USENIX and SAGE dues are spent. Chart 1 shows the total dues income ($746,402 in 1999) divided by type of membership. Chart 2 then presents how those dues are spent. Note that income from our conferences cover all costs of the conference office, exhibition and marketing. Chart 3 shows how the executive office spends its money. The "other" category covers such items as taxes, licenses, bank charges and miscellaneous expenses. Chart 4 indicates where most of the money allocated to Good Works and standards activities are spent ($962,513) in 1999. (See the USENIX Web site at *<http://www.usenix.org/about/goodworks.html>* for a description of our Good Works program). These funds come from the income generated by the USENIX conferences and interest income from the Association's reserve fund.

Two charts deal with SAGE income (chart 5) ($270,143 in 1999) and direct expenses (chart 6) ($185,607).

Allocated expenses (staff and overhead) are not reflected in the direct expenses chart.

Conferences

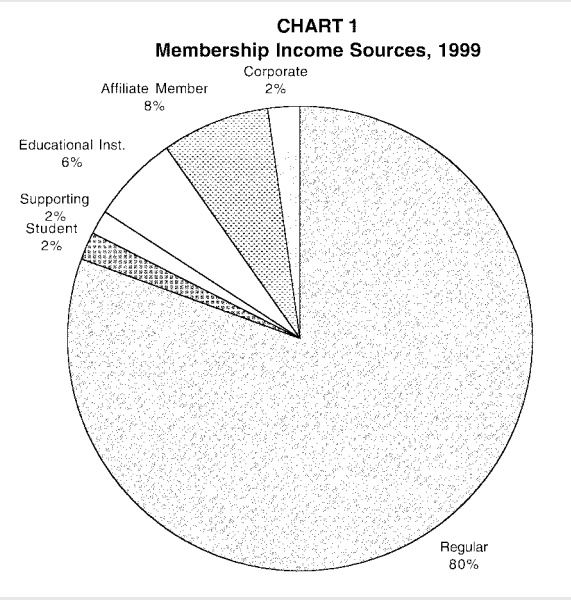
In 1999, USENIX sponsored the most conferences ever: three major conferences and 10 smaller workshops. Over 6,000 people attended these events. Tutorials continue to be popular. We boast some of the finest teachers in the industry and this is evidenced by the increasing number of persons taking them. Conferences and workshops that operated at a net loss were COOTS, Smart Cards, NETA, DSL, and USITS. Revenues exceeded expenses for all of the conferences included in the Annual Technical, LISA, and Security.

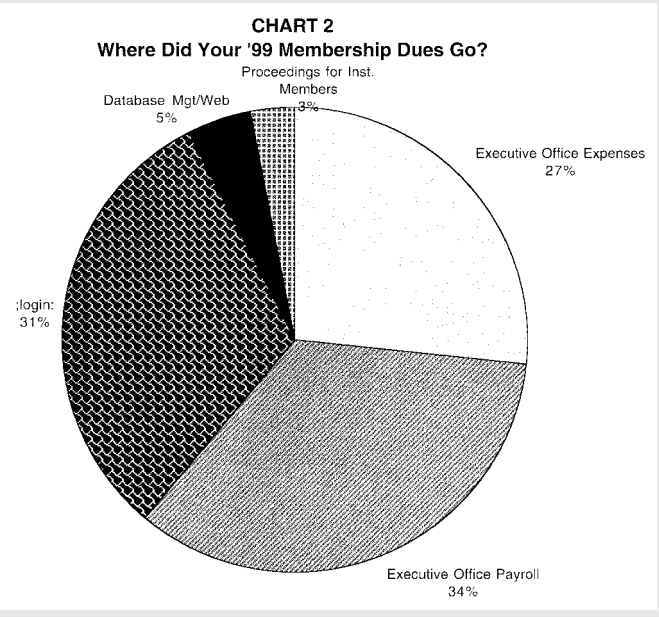
Publications

We published 8 issues of *;login:*. The quality and quantity of material keeps improving.

Projects and Good Works

The Association's healthy year-end budget was supported by strong returns on our investments, which netted $256,644 for our Good Works program. USENIX allocated over $1,000,000 for its Good Works program, and spent nearly all of it. These funds are used to provide stipends to students to attend USENIX and SAGE conferences, scholarships,





support student research, promote outreach to representatives on campuses, as well as several innovative, computing-related projects. Over 360 institutions have been represented in the USENIX Student Stipend Program. To date, over 100 schools have designated outreach representatives. Our Scholastic Program provides funding for scholarships and student research projects.

See *<http://www.usenix.org/about/goodworks.html>* for a complete list of our Good Works program.

CHART 3
Executive Office Expenses, 1999

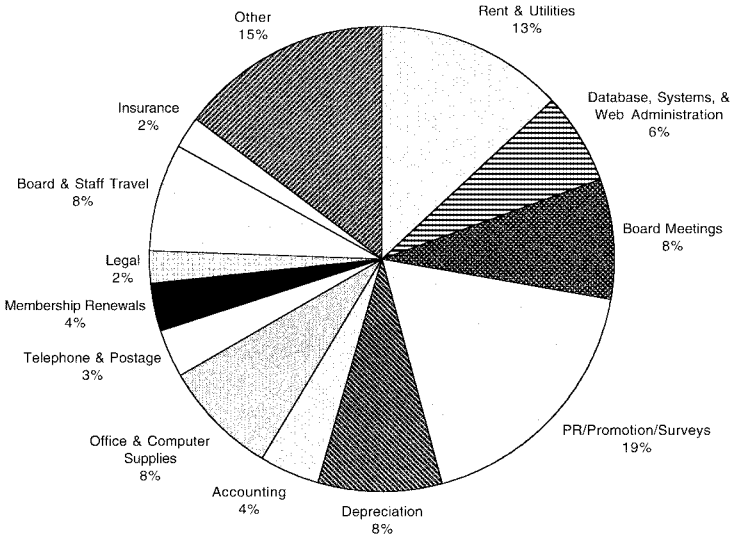


CHART 4
Projects and Good Works, 1999

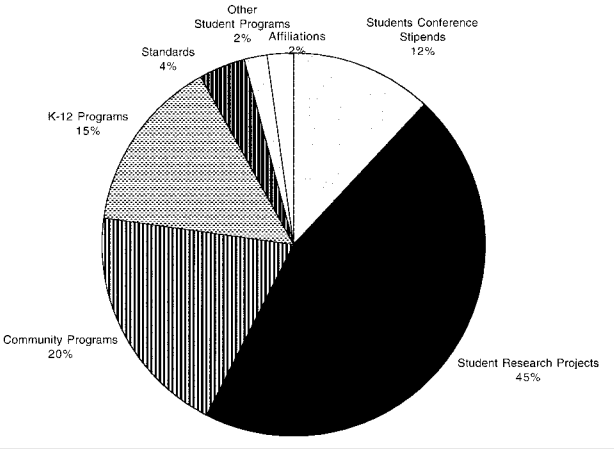


CHART 5
SAGE Income Sources, 1999

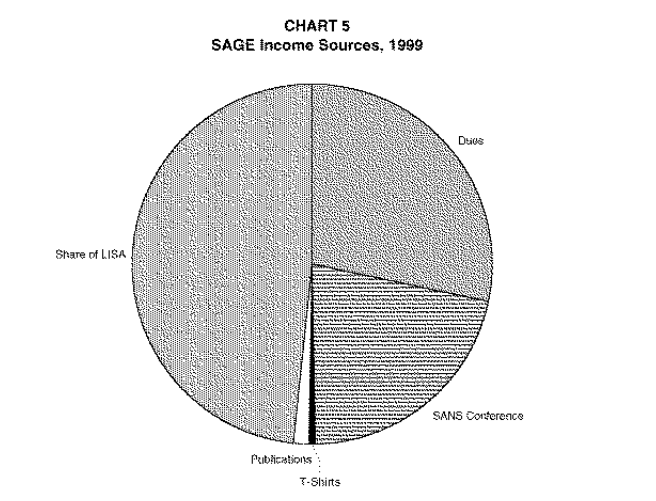
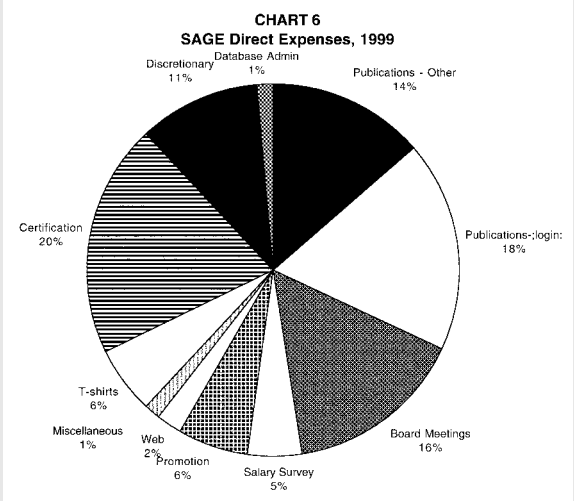


CHART 6
SAGE Direct Expenses, 1999



1999 Financial Statements

STATEMENTS OF FINANCIAL POSITION
AS OF DECEMBER 31, 1999 AND 1998

ASSETS	1999	1998
CURRENT ASSETS:		
Cash and cash equivalents	$2,107,048	$2,079,358
Receivables	115,555	60,714
Prepaid expenses	57,841	112,405
Inventory	18,544	22,311
Total current assets	2,298,988	2,274,788
INVESTMENTS	7,755,283	5,648,278
FURNITURE AND EQUIPMENT	155,028	125,290
Total assets	**$10,209,299**	**$8,048,356**
LIABILITIES AND NET ASSETS		
CURRENT LIABILITIES:		
Accrued expenses	$134,564	$231,428
Total current liabilities	134,564	231,428
COMMITMENTS AND CONTINGENCIES		
UNRESTRICTED NET ASSETS	10,074,735	7,816,928
Total liabilities and net assets	**$10,209,299**	**$8,048,356**

STATEMENTS OF ACTIVITIES FOR THE YEARS ENDED DECEMBER 31, 1999 AND 1998

REVENUE	1999	1998
Conference revenue	$4,102,871	$3,696,590
Workshop revenue	928,746	729,267
Membership dues	746,402	533,129
SAGE membership dues and other income	270,143	190,952
Product sales	47,712	48,616
Total unrestricted revenue	**6,095,874**	**5,198,554**
OPERATING EXPENSES		
Conference expenses	1,954,395	1,766,978
Personnel and related benefits	1,136,834	1,016,021
Projects and good works	962,513	812,167
Workshop expenses	724,596	500,677
Other general and administrative	628,736	545,406
Membership; login:/web	319,472	299,325
SAGE expenses	185,607	129,351
Product expenses	66,360	46,886
Total Expenses	**5,978,513**	**5,116,811**
Net operating surplus	117,361	81,743
NONOPERATING ACTIVITY		
Donations	6,015	
Interest and dividend income	215,943	213,034
Gains and losses on marketable securities	2,011,441	788,667
Investment fees	(86,938)	(43,797)
Net investment income and nonoperating expense	2,140,446	963,919
INCREASE IN NET ASSETS	2,257,807	1,045,662
NET ASSETS, BEGINNING OF YEAR	7,816,928	6,771,266
NET ASSETS, END OF YEAR	**$10,074,735**	**$7,816,928**

MEMBERSHIP

Telephone: 510 528 8649
Email: <*office@usenix.org*>

PUBLICATIONS/WEB SITE

<*http://www.usenix.org*>
Jane-Ellen Long <*jel@usenix.org*>
Telephone: 510 528 8649

USENIX SUPPORTING MEMBERS

Addison-Wesley
Earthlink Network
Edgix
Interhack Corporation
Interliant
JSB Software Technologies
Lucent Technologies
Macmillan Computer Publishing, USA
Microsoft Research
Motorola Australia Software Centre
Nimrod AS
O'Reilly & Associates Inc.
Performance Computing
Sendmail, Inc.
Smart Storage, Inc.
Sun Microsystems, Inc.
Sybase, Inc.
Syntax, Inc.
Taos: The Sys Admin Company
UUNET Technologies, Inc.

STATEMENTS OF CASH FLOWS FOR THE YEARS ENDED DECEMBER 31, 1999 AND 1998

CASH FLOWS FROM OPERATING ACTIVITIES	1999	1998
Increase in net assets	$ 2,257,807	$ 1,045,662
Adjustments to reconcile increase in net assets to net cash provided by operating activities:		
Depreciation	44,498	40,748
Realized and unrealized gains on investments	(2,011,441)	(788,667)
(Increase) decrease in current assets:		
Receivables	(54,841)	(11,817)
Prepaid expenses	54,564	19,591
Inventory	3,767	8,013
Increase (decrease) in liabilities:		
Accrued expenses	(96,864)	107,917
Deferred revenue	—	(200,613)
Total cash provided by operating activities	**197,490**	**220,834**
CASH FLOWS FROM INVESTING ACTIVITIES		
Purchase of investments	(9,628,860)	(2,183,978)
Sale of investments	9,533,296	2,333,141
Purchase of furniture and equipment	(**74,236**)	(**39,645**)
Total cash provided (used) by investing activities	(**169,800**)	**109,518**
INCREASE IN CASH AND CASH EQUIVALENTS	27,690	330,352
CASH AND CASH EQUIVALENTS, BEGINNING OF YEAR	**2,079,358**	**1,749,006**
CASH AND CASH EQUIVALENTS, END OF YEAR	**$ 2,107,048**	**$ 2,079,358**

If This Be Open, Let Us Make the Most of It

by Daniel Geer
President, USENIX Board of Directors

<*geer@usenix.org*>

USENIX is home to a lot of people who look kindly on "open systems." So long as we don't try to define the term too closely, many of us can generally agree on a few things that fall under that category or, dare I say, that way of life. But try to be precise on what constitutes "open" and things tend to fall apart in a way that mimics the sectarian divisions arising in any set of millennial beliefs. This is not news.

What may be, relatively speaking, news is that open systems are no guarantee of perfection even at what they are best at, and what they are best at is harnessing the brains of a larger collection of critics than a profit-making company can ordinarily justify to its investors or its customers. Eric Raymond says, "Given enough eyeballs, all bugs are shallow." This is just the information-age version of "Practice makes perfect" – but as my father would always correct me, the full form is "Practice makes perfect only when you are practicing perfection."

As a specialist in security, I look for perfection when I can get it and risk management when I can't. By now, everyone knows never to trust a cryptographic algorithm until everybody who is any good has taken a run at it. By now, almost everyone has heard that substantial vulnerabilities are nearly never in the crypto part of a security system, but rather in handling hostile triggering of weird-ass error conditions. Within the past year my old friend Kerberos, certainly one of the most open and widely used security systems, turned out to have a buffer overflow vulnerability. There is no better example out there to prove by demonstration that even if there are enough eyeballs to make all bugs shallow, that does not mean that those eyeballs are paying attention. Just like my father said, practice makes perfect only when you are practicing perfection.

This is about rigor, tempered with brutal marketplace realities. It is naive to imagine that sustained rigor is a natural concomitant of volunteer labor. It is wishful to imagine that rigor can come without costs that don't have to be covered by somebody with money to spend, whether that money flows from taxation, gratitude, profit, or charity. To the extent that the systems USENIX folk build and operate are ever more critical to the world as we know it, we cannot imagine that we'll get what we want – systems that can be improved upon because we can understand them in detail – just by wishing it so. It is hard to be one for all and all for one in the glare of an IPO. It will be hard to ignore rigor when certification leads to licensure and thence to malpractice standards.

So here's my challenge to you: That you are a USENIX member makes you atypical. That you are a USENIX member who actually reads this piece makes you more atypical. That you are a USENIX member who reads this piece and has enough spare cycles to coherently think through what it is USENIX can do, if anything, makes you more atypical still. Let me leave you with a problem statement:

How much rigor do systems need to let people like us stay in the driver's seat, and can USENIX help?

USENIX Conference Office Moves

In September, the USENIX Association moved its conference department from Lake Forest into headquarters in Berkeley. The southern California office was closed because Judy DesHarnais, who has been the USENIX meeting planner for the past 20 years, is moving to Hawaii. She will continue to work part-time for USENIX as an Associate Meeting Planner.

Barbara Freel <*barbara@usenix.org*>, has been hired as the new Conference Director. For the past nineteen years, Barbara has been managing large meetings and conventions for several professional associations in the healthcare field and, most recently for EPRI, the Electric Power Research Institute. She earned her Certified Meeting Professional designation in 1987.

Owen Rundall has been hired as an assistant meeting planner. He has just graduated from college, and has nine years of experience in the meetings industry, including the International Society for Magnetic Resonance in Medicine, and the Convention Sales department of the Hilton San Francisco and Towers. Moun Chau, who has been a production editor in the Berkeley office, is helping with registration and other conference activities on an interim basis.

NOTE ADDRESS CHANGE!
Please change all address entries you may have for the USENIX Conferences to:

USENIX Conference Department
2560 Ninth Street, Suite 215,
Berkeley, CA 94710
Phone: 1.510.528.8649
Fax: 1.510.548.5738
Email: conference@usenix.org