inside:

CONFERENCE REPORTS
from the 5th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)

SAGE NEWS AND FEATURES
Securing an NT-Based DNS Server
Elizabeth Zwicky on SNMP
How to Set Up a Time Server
Toolman

STANDARDS REPORTS
BOOK REVIEWS
USENIX NEWS

features:

Ypmake: A Tool for NIS
by Albert Chin-A-Young

Fundamentals of Troubleshooting TCP/IP
by Timothy Hoff

Privacy in the Real World
by Dan Geer

Email Archiving
by Shane B. Milburn

and more...
2nd USENIX Symposium on Internet Technologies and Systems (USITS '99)
Co-sponsored by the IEEE Computer Society Technical Committee on the Internet
October 11-14, 1999
Regal Harvest House Hotel, Boulder, Colorado, USA
Web site: http://www.usenix.org/events/usits99

3rd Annual Atlanta Linux Showcase
Co-sponsored by USENIX, Atlanta Linux Enthusiasts, and Linux International
October 12-16, 1999
Cobb Galleria, Atlanta, Georgia, USA
Web site: http://www.linuxshowcase.org

13th Systems Administration Conference (LISA '99)
Co-sponsored by USENIX & SAGE, The System Administrators Guild
November 7-12, 1999
Seattle Convention Center, Seattle, Washington, USA
Web site: http://www.usenix.org/events/lisa99

February 8-11, 2000
Malmö, Sweden
Web site: http://www.nordu.org/NordU2000/

7th USENIX Tcl/Tk Conference (Tcl/2k)
February 14-18, 2000
Austin, Texas, USA
Web site: http://www.usenix.org/events/tcl2k

Workshop on Applications of Embedded Systems
Co-sponsored by the MIT Media Laboratory
March 20-22, 2000
San Francisco, California
Submissions due: November 15, 1999

SANS 2000—9th Annual System Administration, Networking, and Security Conference
Co-sponsored by the SANS Institute and SAGE
March 21-27, 2000
Orlando, Florida, USA
Web site: http://www.sans.org

SANE 2000—2nd International System Administration and Networking Conference
Organized by NLUUG, co-sponsored by USENIX and Stichting NLnet
May 22-25, 2000
Maastricht, The Netherlands
Web site: http://www.nluug.nl/events/sane2000/
Submissions due: November 1, 1999

USENIX Annual Technical Conference
June 18-23, 2000
San Diego Marriott Hotel & Marina, San Diego, California, USA
Submissions due: November 29, 1999

LISA-NT—3rd Large Installation System Administration of Windows NT/2000 Conference
Co-sponsored by USENIX & SAGE, The System Administrators Guild
July 30-August 2, 2000
Seattle, Washington
Submission proposals due: February 16, 2000

4th USENIX Windows Systems Symposium
August 3-4, 2000
Seattle, Washington
Submission proposals due: February 11, 2000

9th USENIX Security Symposium
Co-sponsored by CERT
August 14-17, 2000
Denver, Colorado, USA
Submissions due: February 10, 2000

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

IN THE FUTURE

2 LETTERS TO THE EDITOR
3 THE USENIX CROSSWORD PUZZLE

CONFERENCE REPORTS
6 Report on the 5th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)

SAGE NEWS AND FEATURES
12 Better than Better
   by Tina Darmohray
13 SAGE AU ’99
   by Hal Miller
14 The Third SAGE!
   by Hal Miller
15 SAGE Certification Update
   by Barbara L. Dijkers
16 System Profiles with Syssumm – A Follow-Up
   by Bruce W. Mohler
18 Securing an NT-Based DNS Server
   by William D. Kramp
20 Managing Network Security with Cengine, Part 2
   by Mark Burgess
29 Toolman: Revision Control Revisited
   by Daniel E. Singer
34 How-To: Set Up a Time Server
   by Hal Miller
38 Enough SNMP to Be Dangerous, Part 3
   by Elizabeth Zwicky

FEATURES
44 Ypmake: A Tool for NIS
   by Albert Chin-A-Young
54 Fundamentals of Troubleshooting TCP/IP
   by Timothy Hoff
58 Privacy in the Real World
   by Dan Geer
62 Email Archiving
   by Shane B. Milburn
65 The Tcsh Spot: Adding Hypertext Links
   and Image Display to htmlview.tcl
   by Clif Flynt
69 IMHO: Y2K
   by Lee Demon
71 FreeBSD: Tracking Stable
   by Rick Leir
74 Musings
   by Rick Farrow

STANDARDS REPORTS
77 The Open Group and IEEE to Develop
   Joint Revision to POSIX and UNIX Standards

BOOK REVIEWS
79 The Bookworm
   by Peter H. Salus

USENIX NEWS
81 2000 USENIX Nominating Committee
   by Evi Nemeth
81 LISA ’99
   by David Parter
82 30 Years Since 1969
   by Peter H. Salus
82 Student Research Grants
   by Gale Berkowitz
89 Meet USENIX New Staff
89 USA Computing Olympiad Picks Final
   Four
   by Don Piele

ANNOUNCEMENTS AND CALLS
92 Workshop on Applications of
   Embedded Systems
96 motd
   by Rob Kolstad

Cover Photo: San Diego Surfers
In the March/April 1992 issue of this journal (then still a newsletter), an announcement appeared on page 3 entitled “Welcome to the New Improved ;login.” The piece, signed “Rob,” marked the beginning of ;login: as we know it today – conference reports, feature articles, “how-to’s,” standards activities, and an expanded book review section. Since that day, editor Rob Kolstad, seeking out and working with contributors, has continually improved ;login:’s quality and interest for you, the membership.

Now it is time for another change. Beginning with the December issue, Rob will take a new position with ;login:, as chair of an advisory Editorial Board (now being formed). The day-to-day editorial responsibilities will be shared by co-editors Tina Darmohray and Rik Farrow. As many of you know, Tina has served as the SAGE News & Features editor since 1995, about the same time that Rik has been writing regular columns and book reviews for ;login:. In addition to their many years in writing and publishing, they own significant expertise in the fields of security and system administration. With an Editorial Board representing other fields, we hope to expand coverage and bring you more issues each year.

We have been fortunate in having Rob serve for so many years as the prime mover in this enterprise. We are doubly fortunate that he has agreed to head the Editorial Board, so that we may continue to benefit from his long experience, his wit, and his wisdom.

This New Improved ;login: will be known as “The Magazine of USENIX and SAGE.” Please let the editors know what you’d like to see in ;login:. And remember, a magazine can only be as good as its writers. Please contribute!

On a more somber note, we have just learned that W. Richard Stevens, noted author of computer books, died on September 1. He was best known for his UNIX Network Programming series (1990, 1998, 1999), Advanced Programming in the UNIX Environment (1992), and TCP/IP Illustrated series (1994, 1995, 1996).

Born in 1951 in Luanshya, Northern Rhodesia (now Zambia), Richard received a B.Sc. in Aerospace Engineering from the University of Michigan in 1973 and an M.S. (1978) and Ph.D. (1982) in Systems Engineering from the University of Arizona. From 1975 until 1982 he was employed at Kitt Peak National Observatory as a computer programmer. From then until 1990 he was Vice President of Computing Services at Health Systems International in New Haven, Conn. After 1990 he pursued a career as an author and consultant.

A testimonial to Richard will be published in the next issue.
More on Certification
From Sergey Babkin
<b>babkin@bellatlantic.net></b>

I was delighted to see the letter from Daniel Brockman published in the August issue of <i>login</i>. I completely share his concerns about the certification of system administrators. I do not work as a system administrator right now but I did in the past and maybe I shall do in the future.

I believe that certification benefits neither the professionals nor their clients. Does it benefit anyone? Of course it does. It benefits two social groups: the bureaucracy that conducts the certification and the people who can’t stay in the business without being protected by a shield of certificate. We can easily find numerous examples in current life.

Take, for example, the Microsoft certification programs. No doubt, Microsoft makes very nice money from selling the materials for a high price and charging thousands of dollars for the certification itself. But that means that the professionals lose this money, and their clients lose this money too because they have to compensate these expenses. Does the presence of certification mean that its owner really knows something about the subject? I doubt it very much. I do not have much respect for the people I know who have this certification. I would not recommend them for any job requiring any intelligence. They are most enthusiastic about getting these certificates.

No wonder they are: if someone can’t prove his knowledge by his job performance, his best strategy would be to hide behind the shield of a certificate or degree. So, if someone needs a person to support Microsoft systems I personally would specifically recommend to not hire anyone with the Microsoft certificate until he or she can prove successful work experience.

Let’s look at automobile mechanics. Have you seen any of the franchised shops along the road, such as Midas, NTB, and so on, boasting about their certified mechanics? Have you had any experience with them? I had and I will never do any business with any of them again. If someone boasts that he is certified by Goodyear, most probably that means that he is a misfit unable to balance wheels after four attempts and when doing that he will scratch the alloy rims with a file instead of using special weights.

I have recently met another example from the area of medicine, a highly certified area. After I moved I went for the first time to a new physician, selected blindly from a list. That was a wrong idea. He seemed to know less about medicine than I do, and I know very little. Well, I won’t go to him any more and I’m quite happy with my new physician, referred to me by a friend.

All the examples I have seen up to now show that mandatory certification does not protect consumers against bad service, and optional certification too often certifies only that the certified shops or persons will provide only bad service.

I also have experience from another side. It happened that I got a Novell NetWare Administration certificate. Does that make me a good NetWare administrator? I doubt it. Attending the courses gave me some interesting knowledge. And I’m probably not a really bad NetWare administrator, at least I have seen a number of worse ones. But that’s not because of this certificate but because UNIX administration and NetWare administration have things in common and most of the time I’m able to figure out or quickly find in the manuals the details I don’t know, based on the basic knowledge I have. And yes, I would recommend the same caution when hiring the bearers of Novell certificates as for Microsoft certificates, or any other certificates, such as CISCO or HP or Oracle.

Based on all this experience, my opinion is: “Certification Considered Harmful.” I also want to comment on another issue. I think that too much attention and space in <i>login</i> and in the activities of USENIX gets dedicated to Windows NT. There are a lot of commercial organizations with Microsoft at the head that support and provide information on Windows NT and I see no reason why my membership payments should support this activity. Yes, I know the argument that many companies have UNIX installed alongside with NT. But I can’t see how it could justify all that attention to NT.

Let’s try to follow this argument to its logical extension. In the ’70s and ’80s many companies had UNIX installed alongside some DEC OSs, such as RT-11, RSX-11, VMS. Did USENIX pay that much attention to these OSes at that time? And now many companies have UNIX installed along with Windows 95/98 and mainframes. So maybe, to be fair, USENIX must also pay as much attention to Windows 95 and OS/390 as to NT? And don’t forget Oracle and Informix; many companies have Oracle and Informix installed on UNIX, so they should be covered too. And throw in VMS, which is not completely dead yet, and OS/400. That brings us to the moment when UNIX would support everything in the world, with UNIX lost somewhere among them (if visible at all). I think that this is a wrong direction, and that USENIX should stop supporting Windows NT.

Shadow Passwords
From Professor Raphael Finkel
<b>raphael@cs.engr.uky.edu></b>

Rik,

I enjoy your Musings column in <i>login</i>. A few times you have mentioned that sysadmins faced with HP-UX tend not to use the shadow passwords because the organization is so nonstandard. We have found a nice way around the problem.
Our shop is pretty diverse, with Irix, Solaris 2, Linux, HP-UX, AIX, and in the past NeXi, Ultrix, and other versions of UNIX. We need to keep various things consistent, in particular, the user community as shown in /etc/passwd. We can't use NIS+, because not all our UNIX variants speak it, and we don't want to use NIS (YP), because it is not very secure. Furthermore, we have a bias against depending on the net for data, because the net or the servers on the net can be unavailable, and we don't want to hang just because an NIS server is out.

Instead, we have a database of users. For each user, we store the encrypted password, the user id, the user name, and which machines the user should have a working account on. The data format is unimportant at this level of discussion. The database is stored on one machine and visible (by NFS, protected in the usual ways against ordinary users) on all. The actual database files could be flat or other; at the moment, they are flat (but in our own format), but we are trying to convert to LDAP.

Whenever it changes, all machines run a Perl script against the data in order to build their own /etc/passwd file. The Perl script takes various actions depending on the UNIX variant. For those that have shadow tables, the script builds the shadow tables, honed to the version of UNIX both in location (AIX likes them in /etc/security/password, but most put them in /etc/shadow) and in format. (AIX puts ! in the password field in /etc/passwd to indicate that the encrypted password is in the shadow file, but HP-UX uses * and others use x; each version of UNIX has a different format for the shadow file itself.) Some versions need some postprocessing. (IRIX needs /sbin/pwconv to be called.)

When we got HP-UX castoff machines for the first time, it was a one-time hassle to upgrade the Perl script to deal with HP-UX, which uses a shadow file per user in /tcb/files/auth. The result was about 50 HP-UX specific lines in the Perl script.

For each user, if the shadow file already exists, it is checked to see if the password has changed; if so, it is changed in place. If the shadow file does not exist, it is built from a template. Shadow files for departed users are removed.

The code may be ugly, but it works fine, and it is maintainable. Adding or deleting users is as simple as editing the database (there is a vi/emacs interface, as well as a batch-mode interface for massive changes). Soon thereafter, the change is seen on all our 150+ diverse machines.

The software is homegrown and freely available (GPL). We have a paper in press on how it all works: Raphael Finkel, Brian Sturgill (Ataman Software, Fort Collins, CO) and Harlan Stenn (PFCS Corporation, Manchester, MO), “Experience with a UNIX System-Administration Tool,” Software Practice and Experience, accepted May 1999. We use the same idea to keep our hosts lists (primarily /etc/hosts) and mounts tables (usually /etc/exports) consistent.


The entire package is called SAT, which stands for System Administration Tools.

To Install GCC . . .

From Max Southall
<max@prninf.com>

Hi Rik,

I installed Solaris7 on several machines, and obtained the GCC compiler from sunsite - now metalab - and it worked just fine.

Enjoyed your ;login: article. I never get to go to conferences - workload too heavy, and company I work for not enlightened enough to pay for training or conferences.

To: max@prninfo.com

From Rik Farrow

Hi Max:

I think the install of GCC would have worked if I had about 1.5 times more disk space in some partitions. But without the additional disk space in three different partitions, it just wasn't going to work. Also, Solaris had to be installed in the first partition (all of my PCs, except the mail server, dual boot). I would have thought that 2GBs was enough to install Solaris and GCC.

Too bad about the conferences. The hall time alone is worth it.
crossword

Across
1. Tells a yarn 60. Belief in unified organic whole
6. Possess
10. Brightly colored fish
14. Great lake
15. Winnie was one
16. Radus's companion
17. Sproge, antistrope
18. Ripped
19. Blue-green color
20. RTF
22. Network protector
24. Slip, slide
26. Can
27. Sleeping place
28. The 13th
29. Elmac or vi, e.g.
32. Imitate
37. Big Ford success
39. Donate
40. Vibratory sound
42. Consumed
43. Painful struggle
44. Shoe string
45. Taker of an oblique course
47. Mischiefous fairy
48. Hebrew coin
50. and the Tramp
51. Distress call
52. Opaque eye part
54. Sweats
56. Joins together

Down
1. Noah's oldest son
2. Caterpillar's second stage
3. Fe
4. Complication
5. More futile
6. Choose
7. Dog word
8. Undershot bucket
9. Waterwheel
10. Be more important
11. Defendant's reply
12. Retentive
13. Auditorium
21. Eye coverings
23. Old age (archaic)
25. Picture for transfer
26. Hebrew unit, 0.1 homer
29. Remove frozen water
31. Yucca-like plant
33. Car feet
34. Rounded convex molding
35. Scuba goals
38. Give back lent money
41. Start the fire again
42. Brick furnace
45. Swindler
46. Yummy cheese
49. Before
53. Step for crossing a fence
55. Garden mollusk
56. Patient, instance
57. Hebrew unit, 0.01 homer
58. Refuse (archaic)
59. Ugly duck
61. Not RUNNING
62. Young Norwegian herring
63. Peat plant
66. Columbus's goal (abbr.)

Solution to the August Puzzle

ODOR: ABACA, HUSK
POPE: PAROL, ANON
TRaversing: FROE
SPHEROID: EFFETE
INN: TRIES
CABLES: PAINTERS
HEELS: WITAN, RHO
ISLE: UHLAN, AVID
NIL: SNEER, STENO
GRAPPLED: SELDOM
DRAIN: VIVA
SCOURS: SORENESS
CONN: TRIDENTATE
ACNE: EATEN, IRON
ROAR: DYERS, CLAD
5th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)

SAN DIEGO, CALIFORNIA
May 3-7, 1999

by Marina Spivak

Mexican influence is clearly evident in San Diego's architecture and food, which is not surprising given that the city was part of Mexico until 1848. San Diego's weather is near-perfect, and so is its list of attractions: from world-famous San Diego Zoo and Sea World, to 22 miles of beaches and rocky shores, to jazz clubs in the historic Gaslamp Quarter – there is something for everyone to enjoy. In May, this list boasted one more world-class attraction: the 5th Conference on Object-Oriented Technologies and Systems (COOTS '99).

Most of the conference technical sessions focused on the increasing demands of distributed computing. Murthy Devarakonda of the IBM T.J. Watson Research Center was this year's program chair. The tutorial program chair was Douglas C. Schmidt of Washington University. Session topics and tutorials focused on design patterns, runtime issues in distributed systems, objects and databases, ORB and mobile-agents optimization, large-scale programming, and Java virtual machine performance evaluations.

This year, a total of 17 papers were selected out of 61 submissions. The Best Student Paper Award went to Uwe Zdun from the University of Essen, Germany, for his research on language support for design patterns. This year the conference seemed cozier than ever, and the number of attendees remained about the same as last year, at 200.

Keynote Address
Summary by Marina Spivak

What was Left Out of Java and Why?
James Gosling, Sun Microsystems, Inc.

James Gosling, the creator of Java, gave an overview of language features that are not present in Java, but whose presence is a subject of many user debates and enhancement requests. The most wanted additions to the language are asserts and parameterized types. Pre/post conditions and invariants go hand in hand with asserts, but there are many questions and subtleties regarding them. Preprocessor macros are still considered evil because they change the software at compile time. Allocation of everything on the heap can be detrimental to performance but makes security easy, since the code is not allowed to corrupt the stack frame. Operator overloading is easy to abuse and difficult to control. Enums are easy to replace with integer constants. Multi-
value returns and in/out values are interesting but not widely required.

Gosling concluded by noting that coping with popularity has been difficult for Java language designers because there are over a million Java developers, resulting in many different opinions. At times it seems that Java is being stretched to meet domain-specific needs such as realtime, scientific, and database. Future changes to the Java language will be dictated by its user community.

Session: Design Patterns

Session chair: Steve Vinoski, IONA Technologies, Inc.

Summary by Irfan Pyarali

Filters as a Language Support for Design Patterns in Object-Oriented Scripting Languages

Gustaf Neumann and Uwe Zdun, University of Essen, Germany

Design patterns are often used to simplify large applications. However, classes and objects that implement these design patterns tend to be scattered throughout the whole system, making patterns hard to locate and maintain.

Uwe Zdun described an object-oriented scripting language, XOTCL (Extended Object TCL), which is equipped with several features that help in the implementation of design patterns. XOTCL allows patterns to be described as instantiable meta-classes. Introspection simplifies adaptation and maintenance, and per-object specialization eases implementation of single objects. Language support for filters allows modification, redirection, and interception of messages delivered to an object.

XOTCL is available for evaluation from <http://nestroy.inf.uni-essen.de/xotcl>.

Performance Patterns: Automated Scenario-Based ORB Performance Evaluation

S. Nimmagadda and C. Liyanaarachchi, University of Kansas; A. Gopinath, Sprint Corporation; D. Niehaus, University of Kansas; A. Kaushal, Sprint Corporation

As CORBA (Common Object Request Broker Architecture) specifications stabilize, the number of available mature CORBA implementations is increasing rapidly. Performance is becoming an important criterion for selecting an ORB. Since the performance of an ORB is greatly influenced by the application context, the operating system, and the underlying network, application developers need to evaluate how candidate ORBs will perform within heterogeneous computing environments. However, the lack of standard and user-extendable performance benchmark suites that exercise all aspects of the ORB endsystem under realistic application scenarios makes performance evaluation a daunting task.

Doug Niehaus introduced Performance Pattern Language (PPL), a higher-level language for describing application-level interaction scenarios. He also talked about Performance Measurement Object (PMO), a test daemon that specifies the creation of CORBA objects, their execution-time behavior, and the relations that hold among the objects.

Both PPL and PMO address the above-mentioned problems in evaluating ORB performance by providing an automated script-based framework within which extensive ORB endsystem performance benchmarks may be efficiently described and automatically executed. Unfortunately, most of the benchmarking results presented by Niehaus were deemed suspect because some of the target ORBs were not compiled with the correct compiler optimizations. More information on this project can be found at <http://ittc.ukans.edu/~niehaus>.

Object-Oriented Pattern-Based Parallel Programming with Automatically Generated Frameworks

Steve MacDonald, Duane Szafron, and Jonathan Schaeffer, University of Alberta, Canada

Steve MacDonald presented the CO2P3S parallel-programming system, which generates frameworks from pattern-template specifications in order to ease complexities of parallel programming. He then introduced phased parallel design patterns that allow specification of time-related aspects in a parallel program. Presented results showed that generated frameworks can be used to quickly implement parallel programs with substantial performance gains over their sequential counterparts.

In light of all the attention design patterns have enjoyed over the recent years, it is interesting to note that their popularity just keeps on growing, but the focus is shifting from their discovery and documentation to development of tools and techniques that make them better integrated into development environments and easier to use — for example, language support, pattern template specifications, and automated code generation. The variety of research and development in support of design patterns is exemplified by the inclusion in this year's technical program of this presentation and those by Uwe Zdun and Lance Tokuda.

October 1999
Session: Runtime Issues
Session chair: Yi-Min Wang, Microsoft Research

Summary by Irfan Pyarali

Intercepting and Instrumenting COM Applications
Galen C. Hunt, Microsoft Research; Michael L. Scott, University of Rochester

Galen Hunt started by explaining Microsoft's Component Object Model (COM). Runtime interception of binary standard interfaces in COM enables the development of an incredible variety of useful component services. Hunt described the implementation of interception in COM, including (a) inline redirection of object-instantiation functions, (b) interception through interface wrappers, (c) tracking of interface ownership, and (d) support for undocumented interfaces. While the techniques described by Hunt were specific for COM, they have relevance to other object models, such as CORBA.

Implementing Causal Logging Using OrbixWeb Interception
Chanathip Namprempre, Jeremy Sussman, and Keith Marzullo, University of California, San Diego

Keith Marzullo presented COPE, a CORBA service based on causal logging. Causal logging is a useful technique for implementing causal consistency because it trades off bandwidth for latency. Applications ignore an action until they have observed that all actions upon which the observed action causally depended have completed.

Interception facilities provided by OrbixWeb filters were used for implementation of COPE. Filters allow invocations in the system to be captured and processed before continuing with the normal flow of the program. Marzullo described problems encountered during implementation of COPE, in the hope that it will help both those who are implementing and those who are planning on using CORBA interception facilities.

Quality of Service Aware Distributed Object Systems
Svend Frølund, Hewlett-Packard Laboratories; Jari Koistinen, Commerce One, Inc.

QoS (Quality of Service) requirements vary vastly among applications. Furthermore, QoS requirements may change as operating conditions and available resources vary. To deliver satisfactory QoS in this context, a system must (a) be QoS-aware so that it can communicate its QoS expectations to external services, (b) trade and negotiate QoS agreements, (c) monitor compliance, and (d) adapt to changes in available resources.

Jari Koistinen presented QML, a QoS specification language that captures the fundamental concepts involved in the specification of QoS properties such as contract type, contract, and profile. Then Jari changed the focus to QRR – the runtime representation of QoS expressions. His description of QRR included discussions of its implementation, representation, programming model, and library support. He concluded that QML and QRR could be considered prototypes for common specification language and interchange format for QoS-enabled distributed systems.

Session: Objects and Databases
Session chair: Rajendra Raj, Morgan Stanley & Company

Summary by Irfan Pyarali

Resource Control for Java Database Extensions
Grzegorz Czajkowski, Tobias Mayr, Praveen Seshadri, and Thorsten von Eicken, Cornell University

Ability to extend object-relational database servers with user-defined functions (UDF) provides flexibility but compromises security. Using Java to implement UDFs alleviates some security concerns, but the problem of resource overconsumption and denial of service that are due to a malicious or buggy UDF is not resolved.

JRes, a Java resource-accounting interface and the subject of this presentation, addresses this problem by (a) detecting and neutralizing denial-of-service attacks, (b) monitoring resource consumption, and (c) on-the-fly query optimization based on the feedback of earlier queries.

The work presented was carried out in the context of Java and an OR database, but the results can be generalized to any Java dynamically extensible execution environment, such as a Web server.

Address Translation Strategies in the Texas Persistent Store
Sheetal V. Kakkad, Somerset Design Center, Motorola; Paul R. Wilson, University of Texas at Austin

Sheetal Kakkad presented Texas, "a portable, high-performance persistent object store. Texas can be used with conventional compilers and operating systems without the need for preprocessing or special operating-system privileges." Texas uses pointer swizzling upon page faults to translate persistent addresses into virtual addresses.

Kakkad described the five primary design issues related to granularity choices that affect persistence implementation: (a) address translation, (b) address mapping, (c) data fetching, (d) data caching, and (e) checkpointing.

Kakkad concluded with some benchmarks but added that further performance measurements are necessary, especially using real applications instead of synthetic benchmarks that do not always model reality very well.
Invited Talk

Summary by Marina Spivak

Safe Yet Efficient Data Sharing in Wide-Area Distributed Systems

Barbara Liskov, Massachusetts Institute of Technology

For this year’s invited talk, Barbara Liskov, Ford Professor of Engineering at MIT, presented techniques used in implementation of Thor, a highly scalable and reliable distributed object-oriented database. The main requirement for Thor was safe sharing: the system must be used in a type-correct way and work properly in the presence of concurrency and failures; that is, it must implement atomic transactions. The challenge for such a system lies in attaining good performance while staying very scalable. Liskov pointed out that the following principles were key to reaching these goals: (1) every detail matters, (2) think small, and (3) be lazy.

Techniques highlighted in the presentation included:

■ use of small pointers, which yield fewer disk accesses and produce better cache utilization;

■ use of a modified object buffer for storage management at servers, which allows write absorption and background I/O;

■ clock-based optimistic lazy concurrency control (CLOCC);

■ hybrid adaptive caching (HAC). HAC is a hybrid of two traditional client caching strategies: object caching, which has high overhead, and page caching, which has low overhead but retains cold objects. In HAC, a whole page is fetched into a page-sized frame; however, when a page is discarded, hot objects from that page are retained.

Professor Liskov concluded the talk by saying that Thor-like persistent object stores provide the basis for distributed computing, and she predicted that they will eventually replace file systems as we know them today.

Information on Thor can be found at <http://www.pmg.ics.mit.edu>.

Session: Optimization

Session chair: Ken Arnold, Sun Microsystems

Summary by Marina Spivak

JMAS: A Java-Based Mobile Actor System for Distributed Parallel Computation

Legand L. Burge III, Howard University; K. M. George, Oklahoma State University

Legand Burge presented Java-Based Mobile Actor System (JMAS), which aims to form a powerful network computing infrastructure from the resources of several interconnected computers, providing the user with the illusion of a single very powerful machine. JMAS utilizes Java technology and a parallel programming paradigm based on mobile agents and the actor message-passing model.

JMAS is the first actor-based solution for large-scale data-intensive distributed applications, which may be interconnected by costly communication links. Locality of reference and resource management – load balancing – are crucial for the performance of such systems: the efficiency of an actor-based computation on a loosely coupled architecture depends on where different actors are placed and the amount of communication traffic among them. JMAS implements a decentralized fault-tolerant load-balancing scheme based on the CPU market strategy, with systems of buyers and sellers.

Adaptation and Specialization for High Performance Mobile Agents

Dong Zhou and Karsten Schwan, Georgia Institute of Technology

Many inefficiencies in mobile-agent programming stem from underlying heterogenous-environment-shielding agent systems, which are frequently based on interpreted languages like Java or Tcl/Tk. Dong Zhou presented two techniques for improving the performance of mobile agents: agent morphing and agent fusion.

Morphing involves changing, invisibly to end users, a runtime agent representation to native code. Techniques like cross-platform binary code generation and access to code repositories can be used to implement morphing. Given that for an application with a large amount of computationally expensive floating-point operations, compiled code implemented with C is almost ten times faster than the Java code realization, morphing can provide significant performance improvements, as well as better predictability of execution.

The intent of agent fusion, a second presented technique, is to remove communication overhead from sets of agents residing and collaborating on one machine. Agent fusion involves compiling collaborating agents as one unit, and, therefore, enabling optimizations across agent boundaries, for example, data copying and thread scheduling/context switching.

Applying Optimization Principle Patterns to Design Real-Time ORBs

Irfan Pyarali, Carlos O’Ryan, Douglas Schmidt, Nanbor Wang, and Vishal Kachroo, Washington University, St. Louis; Aniruddha Gokhale, Lucent Technologies, Bell Labs

Irfan Pyarali described a number of ORB Core and Object Adapter optimizations employed in the implementation of TAO – a high-performance realtime ORB developed in the DOC center at Washington University. Presented optimizations included: (1) perfect hashing and active demuxing for improved predictability and performance in request demultiplexing; (2) POA and stubs bypassing for optimized collocation; (3) Thread Specific Storage for efficient...
memory management; and (4) optional field removal from GIOP protocol overhead.

The key result of this work is that it demonstrates that the ability of CORBA ORBs to support realtime systems is mostly an implementation detail.

TAO source code and information are available at <http://www.cs.wustl.edu/~schmidt/TAO.html>.

**Session: Programming in the Large**

Session chair: Joe Sventek, HP Labs

Summary by Marina Spivak

**The Application of Object-Oriented Design Techniques to the Evolution of the Architecture of a Large Legacy Software System**

Jeff Mason and Emil S. Ochotta, Xilinx Inc.

Most Object-Oriented Analysis and Design (OOAD) methods assume that software developers apply the techniques at the beginning of the design process and continue to use them as software matures; however, many large legacy software systems predate OOAD techniques. Furthermore, since pressures of commercial competition focus directly on adding features, fixing bugs, and releasing product on time, software developers often skimp on things that should be done for long-term benefit. The long-term price is the cost of maintenance. Trying to satisfy the needs of business and software engineering proves to be a difficult trick, and the key is to find the right balance.

Emil Ochotta presented the bumps and triumphs of the first stage of the rearchitecturing effort using OOAD for a large legacy system at Xilinx Inc. The shortcomings of the legacy system included long compilation times, lack of encapsulation and insulation, and complex inter-module dependencies. Emil attributed a large portion of the initial rearchitecturing effort's success to the support from the company's senior management and to the development of key concepts, which helped bridge a large semantic gap between the project goals and actual architectural and code changes.

**Supporting Automatic Configuration of Component-Based Distributed Systems**

Fabio Kon and Roy H. Campbell, University of Illinois at Urbana-Champaign

In many contemporary systems, the graceful failure of one module is not properly detected by other modules, leading to the failure of the whole system. Fabio Kon asserted that explicit representation of intercomponent dependencies would provide the necessary common ground for fault tolerance and automatic dynamic reconfiguration. He then described a prototype infrastructure for representing runtime component dependencies.

In the presented model, each component is managed by a component configurator, which is responsible for storing the dependencies between the component it manages and other components in the system. For example, when a component is being destroyed, its component configurator can notify all of its dependents, allowing them to reconfigure – that is, respond to change – if necessary. In the CORBA implementation of this model, prerequisites for software components can be specified in two ways: (a) in terms of persistent IORs – then the implementation repository can be used to create a new object if one is not available; and (b) in terms of service type attributes – then Trading Service can be used to locate a component satisfying the specified requirements.

Kon concluded the talk with discussions of two application scenarios: (1) using the framework to support on-the-fly reconfiguration of dynamic TAO, a reflective extension of TAO real-time ORB; and (2) using the model in 2K, a fault-tolerant self-adapting distributed operating system. Information about these projects is available from: <http://choices.cs.uiuc.edu/2k> and <choices.cs.uiuc.edu/2k/ComponentConfigurator>.

**Automating Three Modes of Evolution for Object-Oriented Software Architectures**

Lance Tokuda and Don Batory, University of Texas at Austin

Lance Tokuda started the presentation by pointing out that architectural evolution in software is costly yet unavoidable, and that one way to reduce costs is to use automation whenever possible. The goal of his work is to promote refactoring research: to evaluate refactorings of large applications and to demonstrate that common forms of architectural change can be automated. Refactorings are behavior-preserving program transformations, which directly aid in the implementation of new architectures.

Refactorings are superior to hand coding: they allow evolution to occur at the level of a class diagram and leave the code details to automation. (Refactorings address the need to evolve from simple to complex designs by automating many common design transitions.) Since "patterns have costs (indirection, complexity)," the design should be "as flexible as needed, not as flexible as possible." Instead of overdesigning, one can start with a simpler structure and, for example, with the help of refactorings, add Bridge, Strategy, and Decorator patterns later, as needed. However, the ability of refactorings to affect algorithms is limited.

Tokuda concluded by saying that the limiting factor in widespread acceptance of refactorings seems to be availability of production-quality tools, and that work in progress includes implementation issues for C++ refactoring tools.
Session: Java Internet Measurements
Session chair: Werner Vogels, Cornell University

Summary by Marina Spivak

The Design and Implementation of Guaraná
Alexandre Oliva and Luiz Eduardo Buzato, Universidade Estadual de Campinas, Brazil

Alexandre Oliva talked about the design and implementation of Guaraná, an extension to Kaffe Open Java Virtual Machine. Guaraná is a reflective architecture that uses the Composite pattern to define meta-configurations. Existing reflective architectures induce developers to create meta-objects that either are tightly coupled or attempt to implement many management aspects of the application "all-in-one." This motivated the development of Guaraná, which improves reuse of meta-level code by defining a meta-object interface that eases flexible composition. It allows meta-objects to be combined through the use of composers. A composer is a meta-object that delegates operations and results to multiple meta-objects, then composes their replies in its own replies. Composers are the glue, and they encourage the separation of the structure of the meta-level from the implementation of individual management aspects.

More information on Guaraná is available from <http://www.dcc.unicamp.br/~oliva>.

Tuning Branch Predictors to Support Virtual Method Invocation in Java
N. Vijaykrishnan, Pennsylvania State University; N. Ranganathan, University of Texas at El Paso

Virtual method invocation, which causes indirect branch execution, has been identified as one of the major bottlenecks for Java performance; furthermore, the proportion of virtual methods in Java programs is likely to increase because of the trend toward fine-grained object design. Accurate target prediction for indirect branches that occur because of virtual methods is critical to the performance of Java applications on deeply pipelined, wide-issue processors: speculative execution is used in such architectures to avoid performance loss associated with branch instructions.

N. Vijaykrishnan presented the results of the study in the use of path history of virtual calls — that is, target addresses of recently executed branches — to accurately predict the target of virtual method invocations and the impact various parameters have on the accuracy of prediction. The following parameters have been varied: number of history buffers, path history length, number of bits of each target address registered in the history buffer, hashing function, and structure of the target buffer.

The XOR hashing scheme with a global path history and a 2-bit update policy performed best for almost all the configurations. These results show that execution of Java code will benefit from more sophisticated branch-predictors.

Comprehensive Profiling Support in the Java Virtual Machine
Sheng Liang and Deepa Viswanathan, Sun Microsystems Inc.

Sheng Liang advocated the use of a general-purpose Java Virtual Machine profiler interface (binary function-call interface) instead of embedding direct profiling support into a virtual-machine implementation. This would make it possible for vendors to ship profilers that work with any virtual machine that implements the interface.

Liang presented such a general-purpose interface, which is efficient and powerful enough to suit the needs of a wide variety of virtual-machine implementations and profiler front ends. He also demonstrated how their approach supports interactive profiling with minimum overhead. Users can selectively enable or disable different types of profiling while an application is running.

Sheng also introduced an algorithm that obtains accurate CPU-time profiles in a multithreaded execution environment with minimum overhead. Since modern OSes do not provide ways to obtain accurate per-thread CPU time, the solution is to determine whether a thread has run in a sampling interval by checking whether its register set has changed.

CLOSING REMARKS

Summary by Marina Spivak
Murthy Devarakonda concluded the proceedings by thanking the attendees, the program committee, and the USENIX staff. Overall, COOTS '99 was a great conference with lots of novel and interesting papers and presentations. We look forward to seeing you at COOTS in early 2001.
Better than Better

by Tina Darmohray

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

Have you ever felt a little sick as you looked at code that implements some critical aspect of your business – because it looks like noodle soup, and you were hoping for something sleek and clean, like the stainless-steel kitchen it was created in, instead? I’ve seen my share of these homergrown tools that are ancient creations with decades of feature-creep and no remaining engineers to tell the story of why it was originally created this way or that. Even the comments are a sad state of affairs, although entertaining, in their own way; the obviously earlier ones sound authoritative and confident, whereas the more recent ones have a tone of submission: “added to work around section B.”

The original code, which has had years to ferment, probably implemented some fabulously necessary feature, maybe even in some straightforward way. But, over the years, it’s been built on and added to by many other talented and clever individuals, all adding a feature here and a workaround there. Over time, these creations become huge monolithic examples of how not to design modular, nimble, easily supported code. In some cases, enough time has passed that some features cease to work, and still others continue to, each without anyone still on board who can remember the purpose of either.

I don’t believe anyone sets out to create these beasts that ultimately take on lives of their own, but, sadly, they’re in no danger of extinction. Many sites suffer from one form or another of them. One day they’re a good idea, the next they’re behemoths causing more trouble that they’re worth. If only there was an easy answer, but as awful as it is living with ‘em, many sites fear they can’t live without them, either. It’s sometimes hard to make the big decision to take on that long-overdue ground-up redesign.

The two most egregious examples I have of these feature-full monstrosities both revolve around mail configuration. (Just for the record, I think this speaks to the fact that I get called for mail messes, more than it does that mail is always a mess!) It’s not hard to figure out why sites are reluctant to do a ground-up redesign; I always tell folks that mail administrators don’t need pagers, since users hunt you down and find you within moments of mail-system failures.

It feels like it’s easier just to patch the existing system than to tackle the fundamental problem. So it ultimately worsens. In one case, I saw a sendmail.cf file that was over 23 pages long. (Contrast that with a more average length of nine or so.) This file was truly spectacular. The blade salesman would accurately describe it as “it slices, it dices, it makes julienne fries.” It did everything, and at the same time, it was unwieldy and unnecessarily complicated.

When I first saw it, I immediately (and maybe somewhat naïvely) suggested just using the standard sendmail.cf files currently available. Sadly, this site suffered from the NIH syndrome, and it became clear I would spend my time arguing with them rather than fixing things, so I added another feature, since so many others had, and went on my way. I just heard recently that they finally took my now three-year-old advice and switched over to a standard file. Their stuff works just as well, and it’s a lot easier to support, now that it’s more in line with what most of the rest of the world does. True, they no longer have a souped-up .cf file that was a coding marvel, but they still have something that is lean and mean and worthy of admiration in its own right.
The other mail scenario I remember wasn’t the .cf file, but was code modification to sendmail itself. The modifications were originally done because sendmail was missing some required features. Over the years, the programmers who modified the code eventually moved on, and support of the proprietary code became increasingly difficult. Ultimately, new sendmail capabilities were able to provide the site with the same or similar functionality and enable them to move to the stock sendmail. By doing so they eliminated their precarious position of relying on an unsupported code base and enabled themselves to take advantage of some of the additional new features of the new off-the-shelf software. As with the other site, mail still flows, but there is a lot less gnashing of teeth involved to keep it that way.

I guess I’m not fancy, when it comes down to it. Or maybe I just lack vision. But I stand by using standards when you’re talking about the backbone for your infrastructure, your services, and your servers. In the long run I think you’ll run a tighter ship, with a lot less effort. Call me boring, but sometimes “standard” is just better than “better.”

---

**SAGE-AU ‘99**

by Hal Miller

Hal Miller is president of the SAGE STG Executive Committee.

<halm@usenix.org>

I had the honor to attend the 1999 annual SAGE-AU conference in July. The format is very similar to LISA, the quality is extremely high, and I’ll be quite honest—I’m very proud of the fact that a SAGE organization outside of SAGE in the US is able to put on such a show. This was the seventh consecutive annual event in Australia, which says something too.

This year’s conference took place at the Novotel Brighton Beach hotel, in Sydney. There were more than 150 attendees and three days of tutorials, followed by two “conference” session days. A very small vendor exhibit took place in the lobby outside the main session room.

SAGE-AU is an independent, not-for-profit agency. Its members are the shareholders, and the group is thus required to put on an annual general meeting, elect officers, and do other corporate-type business. That meeting took place at the end of the first conference day. The chair of each of the state “regional groups” (roughly equivalent to US “local groups”) presented a status report, all of which pointed at the strength of the overall organization.

Recently, the Australian federal government undertook to legislate away bad things on the Internet, much as the US Congress did with the Communications Decency Act of 1996. Parliament passed into law a set of rules, which provide lots of penalties, that prohibit the hosting of pornography on Australian machines. The computing societies are, quite understandably when you read the legislation, up in arms, especially as none was consulted. SAGE-AU sponsored a panel session at this conference, attended by the public, including some news media (the largest national paper and others). The panel consisted of one senator, representatives of other senators, ministers, and various agencies, all from both sides of the question. The session was too short really to accomplish much, but it did establish in the eyes of some high-ranking government officials the fact that a professional body of system administrators exists right there in Australia that is available to them as unpaid consultants for this type of activity. One can only hope that much good will result from that notice, and the indications are that this may be realized.
Tutorials included:

Geoff Halpin and Elizabeth Zwicky (count on seeing this one at LISA – it went over big-time!) on Management 101, covering “soft skills”

Richard Sharpe on SAMBA implementation

Greg Rose (in his home city) on Real World Applications of Cryptography

Geoff Halpin again, on Auditing, the Unexpected Ally

Peter Galvin (the third Yank at the conference) on Basics of UNIX Security

Peter Samuel, Q-Mail implementation

Anthony Baxter, Python introduction

Mike Ciavarella with another one I’d really like to see come to LISA, Documentation Techniques for System Administrators

Peter Galvin again, Advanced Solaris System Administration Topics (no surprise there)

Elizabeth Zwicky with more soft skills, Evaluating a Site’s System Administration Maturity

Andrew van der Stock, Securing BackOffice for E-Commerce Applications

The invited talks:

Peter Elford, of Cisco Australia and one of the two or three most key individuals in the design and implementation of AARNet, the Australian Academic and Research Network (Internet) while I lived there, gave the keynote. He spoke about the recent history of telco carrier development, voice versus data growth rates, and anticipated growth directions. He says that the trend is changing from carriage of data on voice-designed networks to carriage of voice on data-designed networks. While it was a very entertaining and enlightening talk, the one line I think most of us will remember longest was: “ATM is like a duck – it can fly, it can swim, it can walk, but it can’t do any of them particularly well.”

Duane Schuitz reported on a new large-scale, scalable Web-server design, using an NFS back end and front-end processor machines.

Luke Mewburne gave an introduction to CVS, including explanations of the results you get from commands, and tips on what the man pages really mean.

David Burren described his low-budget virtual network, consisting of a PC at each end, that runs ssh over PPP over IP rather than a modem.

Catherine Allen gave a series of tips for how to prepare your site for a security audit.

Andrew van der Stock presented a list covering how to secure an NT Back Office box for e-commerce. He made the interesting point that, regardless of the operating system you use, it pays to use NTP to ensure your log entries are synchronized, just in case you end up in court.

Peter Galvin gave the second keynote, explaining “firefighting” and why it happens to us, and giving some tips for controlling it.

William Shipway and Christian Kraus reviewed the design and development history of a 1,600-lawyer national firm’s migration from Mac/UNIX to NT.

Chris Miles gave a follow-on from last year’s paper, discussing implementation of a toolkit for monitoring, messaging, and notification.

Jeff Alexander of Microsoft Australia discussed Windows 2000 security.

I gave a paper on the problems of growing a site to petabytes of storage.

Elizabeth Zwicky returned with a discussion of use of NT in a firewall environment.

Rex Walters of Network Appliance discussed where they’re going in the area of metadata issues.

Gordon Rowell gave a how-to on DNS implementation.

Richard Sharpe wrapped it up with a discussion on network debugging.

I hope to see more people traveling back and forth between the various SAGE-* conferences in the future. The quality is top-notch, and it gives each of us an opportunity for more venues to present papers and to listen to more. Perhaps we will some day be able to better coordinate “themes” among the various conferences, and all of us will be able to select the one(s) that best suit our personal needs. In any event, the sum total of benefit to our profession continues to climb.

The Third SAGE!

by Hal Miller

<ehalm@usenix.org>

Once again a regional group of system administrators has banded together to formally create a guild. I am thoroughly delighted to announce the formation of SAGE-WISE (Wales, Ireland, Scotland, and England)! The interim president is Anna Langley, vice president is Nigel Barker, secretary is Pete Humble, and treasurer is Jonathan Crompton. Ordinary committee members are Jon Morgan and Anthony Botham.

There have been many attempts to form other SAGE units, including ones in the UK. Most of them died in the discussion phase, with only a few people involved. This one not only has more than four dozen members already, but also has a mailing list that is already repeatedly proving useful as a technical resource to UK members. The enthusiasm has stretched not only across the Atlantic to North America, but to Australia and New Zealand as well!

Reciprocal member-benefit arrangements will be worked out shortly among all SAGEs.

On behalf of SAGE, I get to say, “Welcome to the club, SAGE-WISE!”
HumRRO developed a draft list of tasks system administrators perform and the knowledge bases, skills, and abilities (KSA) required to perform those tasks. Four focus groups were conducted to review and hone that list: two during SANS in May, one in the Bay Area, and one in Chicago. In addition, one-on-one interviews were conducted with five members of the SAGE certification advisory council to provide additional feedback and a good confidence level in the list.

The task and KSA list has been incorporated into a system administrator occupational analysis survey. The survey will ask respondents to rate the relative importance of each task and KSA and the level of the KSA required to perform their duties. In addition to the task and KSA list, the survey asks respondents for demographic information and other data related to the certification project. The draft survey was posted to the certification advisory council for comment and was reviewed in detail by the certification subcommittee. The advisory council participated in a pilot of the survey prior to public posting.

This system administrator occupational analysis survey should be available on the SAGE certification Web site in October (<http://www.usenix.org/sage/cert/>). The survey will take someone approximately 45 minutes to complete and submit. The availability of the survey is being publicized widely by SAGE in order to encourage as much response as possible. Please take the time to complete and submit the survey yourself, and also tell all your peers about the survey, especially those who may not be SAGE members or are new system administrators. The survey will be available for a posted period of time to collect responses.

When the survey posting period is concluded, HumRRO will analyze the results. The results will be incorporated into subsequent phases of the certification project. Preliminary results should be available at the LISA conference in Seattle in the results session of the technical program and the SAGE community meeting (BoF).

**Sponsors**

There is now a sponsorship program for the SAGE certification project. The purpose of sponsorship is to develop recognition for the project and acquire funding to support it. Details about sponsorship are available on the Web site. Financial contributions are not required for listing as a sponsor. Please encourage your employer and/or your training provider to be a sponsor.

**Other Developments**

The Department of Defense (DOD) has issued a mandate that all system administrators will require "level 1" certification. The certification is required by December 31, 1999, for those working on classified systems and December 31, 2000, for those working on nonclassified systems. While the full details of the requirements are not yet available to SAGE, current information suggests that they are limited to security issues. The SANS Institute is developing a certification program specifically to meet the DOD security requirements.

The Linux Professional Institute is in full swing in their certification program, to be ready in the third quarter of 1999.
system profiles with syssumm - a follow-up

My article in the April 1999 ;login: on “System Profiles with Syssumm” described a series of Perl scripts developed as a “proof of concept,” the goal being to create profiles of a large number of diverse computer systems and display the results via a Web browser. In that article, I asked anyone interested in collaborating on such a project to contact me via email. This article, written six months after the first one, summarizes what has happened to syssumm.

There are 60 people on the syssumm mailing list and eight active developers. The main mailing list represents people from Australia, Belgium, Brazil, Canada, Germany, Japan, New Zealand, Norway, Sweden, Switzerland, the United Kingdom, and the United States.

The following people have directly contributed to the source code of the syssumm project: Martin Andrews, Jeffrey W. Collyer, Frank Crawford, Paul Farrell, Brett M. Hogden, Keong Lim, Bruce Mohler, Jeff Putsch, and Bart Swennen.

How Has Syssumm Changed?

At the time of the original article, syssumm supported HP-UX 9.X and 10.X and Linux (Red Hat 5.1). At the present time, it supports:

- AIX 4.X
- IRIX 5.3 through 6.5
- HP-UX 10.X and 11.X
- HP-UX 9.X (partial)
- Linux (Red Hat 5.1 - 6.0)
- SunOS (Solaris 2.X)

We’re hoping for modules soon for DEC/UNIX (Tru64) and Windows NT.

The original Web interface offered the choice of displaying a specific system’s profile or listing all available profiles. Paul Farrell added functionality that allows searching on a combination of vendor (e.g., HP, Sun), operating system (e.g., HP-UX 10, RH Linux 6.0), location, and organization. (See the figure on p. 17.) The values in the lists represent the values in the currently stored profiles on your Web server.

What Are the Near Future Development Plans for Syssumm?

We want to complete the modules for DEC/UNIX (Tru64) and Windows NT, and we’re adding code to identify a Web server running on the remote system.

How Are People Using Syssumm?

Syssumm solves a number of problems for system administrators:

- It bundled a number of other administrative procedures and made some of them available electronically that weren’t before. (Bart Swennen)

Managers can get server hardware, software, and configuration information directly without going through a system administrator. I’ve had a couple of managers call me since I’ve installed the software. . . . When I told them they could get the info by going to our Web page, they were impressed. (Paul Farrell)

We now have detailed on-line configuration information for each system. [syssumm] also gives me a place to hook in more information as needed. (Jeff Putsch)
Having all of the config information gathered in one place, formatted, and up-to-date is a big time saver. All I need to do to get a quick snapshot of a server is click on an HTML link.

(Paul Farrell)

The information is helpful in planning upgrades, changes, etc. Like most sites, we don’t throw out good equipment, we just shuffle it between machines (particularly disks). Having all the details at hand saves considerable time trying in planning such rearrangements, as all the information is up to date.

(Frank Crawford)

One of the managers said to me, “We got all of this, and it didn’t cost us anything?” It was definitely a big PR moment for Open Source software around here.  

(Frank Crawford)

Also as a by-product, the mailing of differences on reboot allows me to keep track of major changes to systems.

(Frank Crawford)

Why Are the Developers Participating?
The nice feeling about participating in such a project is that you feel you can really contribute to something.

(Bart Swennen)

I also feel like I’m giving something back to the UNIX community, which makes me feel good. I’m not a superhacker, but you don’t have to be to contribute to something like this.

(Paul Farrell)

I get something useful out of it and hopefully have contributed useful stuff back.

(Jeff Putsch)

I have learnt many new commands or arguments to existing commands as a means of digging deeper into the system details.

(Frank Crawford)

The syssumm developers are always looking for more people to contribute code and to test it. If you’re interested in collaborating on this software or if you’d just like to keep track of the progress of the software and use it once it’s more functional, please contact me at <bruce.w.mohler@saic.com>.

Conclusion
Syssumm allows you to profile your diverse UNIX systems (and, soon, your NT systems too).

It’s extensible, so you can add code for the information that may be unique to your site (and then contribute it back to the project!).

The format of the Web pages is under your local control. If you need to see categories and subcategories in a different order, you merely edit a file on your Web server.

Syssumm is free, released under the Gnu Public License. You are welcome to copy the software so long as you attribute authorship and don’t charge more than it costs you to copy it.


October 1999
securing an NT-based DNS server

With the shift from Novell to Windows NT to provide file- and print-sharing to our users, it was decided that NT might be an option for providing DNS services as well. We soon found that many services beyond DNS were accessible to the outside world. By combining trial and error with monitoring the network, we stripped down the NT box to its bare essentials for providing DNS services.

Console or Remote Management

Beyond using the command line to manage the DNS service on each system, there are two ways of securely managing the Microsoft NT DNS servers using the GUI interface. Either run the windows-based GUI manager locally to each server, or allow remote management from an internal, secure network. Local access only allows access from the console of each server. This is okay for one or two servers, but can be difficult with more servers. The other option is to configure an internal network that the DNS servers can communicate on. All the primary zone information can reside on an internal server that propagates the information out to secondary DNS servers. The secondary then provides DNS services to external networks through a second interface.

The first step is to have only the network protocol TCP/IP loaded. Under the Control Panel, select the Network icon. Click on the Protocols tab and remove any protocols that are listed other than TCP/IP. The next step is based on your need either to have only local, GUI-based access from the console for each DNS server, or have distributed control from an internal, secure network.

Local Console Only

To permit only local, GUI-based access to the DNS service running on that individual system, the Microsoft Loopback adapter needs to be installed. The loopback is needed to allow access to ports 135 and 1028, which are used by the Microsoft DNS GUI interface. These two ports will be blocked on the external interface. Click on the Adapter tab and select MS Loopback Adapter from the menu of choices. Select the frame type of 802.3.

Remote Management

If distributed access is required, a second network device will need to be installed in the system that is attached to an internal network. It is preferable that the internal network be isolated from other systems at the site, but it can use a shared network with other servers and clients. If access over the Internet is required for management of the DNS servers, an encrypted tunnel should be used to access the internal network.

Each interface needs to have a unique IP address. At the Microsoft TCP/IP Properties window, click on the Routing tab, and make sure that IP Forwarding is not enabled. Then click on the IP Address tab. For the loopback adapter, or the internal interface, enter a legitimate IP address for your network, or use one of the reserved addresses like 192.168.100.1, from the RFC 1597. It specifies private addresses in the following ranges: 192.168.0.0–192.168.255.255, 172.16.0.0–172.31.255.255, and 10.0.0.0–10.255.255.255.

Filtering Ports

The next step is to filter the IP access to the ports on the DNS server network interfaces. While still viewing the TCP/IP properties, click on the Advanced button. Click on the Enable Security check box, then click on the Configure button. Within the TCP/IP Security window, select the adapter that is used for the external interface. By default, all
TCP and UDP ports are enabled, as well as all IP Protocols. For TCP and UDP, click on the Permit Only button for each and add port 53. This will allow remote access to the DNS server but will block access to any other services using other ports. Now select the internal or loopback interface and click on the Permit Only button for each. For the TCP connections, add ports 53, 135, and 1028. Port 53 will need to be added for UDP. TCP ports 135 and 1028 will allow the Microsoft GUI-based DNS manager access to the DNS service. Allowing open access to all ports on the internal interfaces would work, but it is safer to keep things as locked down as possible.

**Shutting Down NT Services**

To limit the vulnerabilities of the DNS server and increase performance, services that are not needed for the operation of the DNS server should be stopped and disabled. Find the Services icon under the Control Panel and select it. Make sure the following services are stopped and disabled: Alerter, Clipbook Server, Computer Browser, DHCP Client, Directory Replicator, Messenger, Net Logon, Network DDE, Network DDE DSDM, RPC Locator, Schedule, Spooler, TCP/IP NetBIOS Helper, and Telephony Service. The License Logging Service is also not needed for the operation of the DNS server, but I am not sure about the legalities of turning it off. It is recommended that each service be set to disabled, not just to manual. If a service is set to manual, another service could start the service up without your knowledge. Some of the services left running should be: Eventlog, DNS Server, NT LM Security Support Provider, Plug and Play, Protected Storage, Remote Procedure Call (RPC) Service, Server, and Workstation. There may be other services running as well, depending on the hardware configuration. These will need to be evaluated as to their risk and in consideration of your site's individual requirements.

**Known Problems**

Some problems I have seen involve the Enable Security box within the Advanced IP Addressing window not staying enabled. This has the effect of turning off any port filtering. One quick fix that sometimes works is to change the IP addresses of the interfaces, reboot, and then change them back to the original IP addresses. If that doesn't work, reloading the service patch should fix it.

**Zone Transfers**

The primary DNS servers should be configured to allow zone transfers only to approved secondary DNS servers. This has the side benefit of automatically updating the secondary when a change is made. The secondary will not have to wait till the refresh time expires before updating its zones. Details of restricting zone transfers and supporting DNS under NT can be found in the O'Reilly book *DNS on Windows NT* by Paul Albitz, Matt Larson, and Cricket Liu.

**Extra Security**

An extra layer of protection for the DNS servers would be to use a firewall or filtering router to restrict accesses to port 53 of the servers. All appropriate service patches and relevant hot fixes should be installed as well. The Windows NT operating system should be further secured to prevent security breaches. A good source of information is available from the SANS Institute's document "Windows NT Security Step by Step" at [http://www.sans.org](http://www.sans.org), and from Microsoft's TechNet.
managing network security with cfengine, part 2

by Mark Burgess

Mark is associate professor at Oslo College and is the author of cfengine and winner of the best paper award at USA 1999.

<Mark.Burgess@iu.hioslo.no>

The point of cfengine is normally to have only one global configuration for every host. This needs to be distributed somehow, which means that hosts must collect this file from a remote server. This in turn means that you must trust the host, which has the master copy of the cfengine configuration file.

The beginning of security is correct host configuration. Even if you have a firewall shielding you from outside intrusion, an incorrectly configured host is a security risk. Host configuration is what cfengine is about, so we could easily write a book on this. Rather than reiterating the extensive documentation, let's just consider a few examples that address actual problems.

A cfengine configuration file is composed of objects with the following syntax (see the cfengine documentation):

```plaintext
class-type:
  classes-of-host-this-applies-to:
    Actual rule 1
    Actual rule 2 ...
```

The rule-types include checking file permissions, editing text files, disabling (renaming and removing permissions to) files, controlled execution of scripts, and a variety of other things relating to host configuration. Some of the "control" rules are simply flags that switch on complex ("smart") behavior. Every cfengine program needs an action-sequence that tells it the order in which bulk configuration operations should be evaluated. For example:

```plaintext
control:
  action-sequence = ( netconfig copy processes editfiles )
```

Now, using Solaris and GNU/Linux as example operating systems, I'll step through some basic idioms that can repeated in different contexts.

Disabling and Replacing Software
One of the simplest things we are constantly asked to do is to disable dangerous programs as bugs are discovered. CERT security warnings frequently warn about programs with flaws that can compromise a system. In cfengine, disabling a file means renaming it to "*.cf-disabled" and setting its permission to 400.

```plaintext
disable:
  #
  # CERT security patches
  #

solaris::

  /usr/openwin/bin/kcms_calibrate
  /usr/openwin/bin/kcms_configure
  /usr/bin/admintool
  /etc/rc2.d/S99dtlogin
  /usr/lib/expreserve
```
Although this is a trivial matter, the fact that it is automated means that cfengine is checking for this all the time. As long as a host is up and running (connected to the network or not), cfengine will be ensuring that the named file is not present.

Another issue is replacing standard vendor programs with drop-in replacements. For example, most sysadmins like to replace vendor sendmail with the latest update from Eric Allman’s site. One way to do this is to compile the new sendmail into a special directory, separate from vendor files, and then to symbolically link the new program into place.

```
links:

solaris::
/usr/lib/sendmail ->! /usr/local/lib/mail/bin/sendmail-8.9.3
/usr/sbin/sendmail ->! /usr/local/lib/mail/bin/sendmail-8.9.3
/etc/mail/sendmail.cf ->! /usr/local/lib/mail/etc/sendmail.cf
```

The exclamation marks mean (by analogy with the `csh`) that existing file objects should be replaced by links to the named files. Again, the integrity of these links is tested every time cfengine runs. If the object `/usr/lib/sendmail` is not a link to the named file, the old file is moved and a link is made. If the link is okay, nothing happens. After putting the new sendmail in place, you will need to make sure that the restricted shell configuration is in order.

```
# Sendmail, restricted shell needs these links
# solaris::
# most of these will only be run on the MailHost
# but flist (procmail) is run during sending...
/usr/adm/sm.bin/vacation -> /usr/ucb/vacation
/usr/adm/sm.bin/flist -> /home/listmgr/.bin/flist
```

```
linux::
/usr/adm/sm.bin/vacation -> /usr/bin/vacation
```

Link management is a particularly useful feature of cfengine. By putting links (actually all system modifications) into the cfengine configuration and never doing anything by hand, you build up a system that is robust to reinstallation. If you lose your host, you just have to run cfengine once or twice to reconstruct it.

Of course, the fundamental tenet of security is to be able to restrict privilege to resources. We therefore need to check the permissions on files. For instance, a recent CERT advisory warned of problems with some free UNIX mount commands that were setuid root. If we suppose there is a group of hosts called “securehosts” that we don’t need to worry about, then we could remove the setuid bits on all other hosts as follows:

```
files:

!securehosts.linux::
/bin/mount mode=555 owner=root action=fixall
/bin/umount mode=555 owner=root action=fixall
securehosts.linux::
/bin/mount m=6555 o=root action=fixall
/bin/umount m=6555 o=root action=fixall
```
One area where cfengine excels over other tools is its ASCII-file-editing abilities. Editing text files in a nondestructive way is such an important operation that once you've used it you will wonder how you ever managed without it! Here are some simple but real examples of how file editing can be used.

```
editfiles:
  # sun4, who are they kidding?
  { /etc/hosts.equiv
    HashCommentLinesContaining '^
  }
  #
  # CERT security patch for void vulnerability
  #
  sunos_5_4::
    { /etc/rmmount.conf
      HashCommentLinesContaining 'action cdrom'
      HashCommentLinesContaining 'action floppy'
    }
```

TCP wrapper configuration can be managed easily by maintaining a pair of master files on a trusted host. Files of the form

```
# /etc/hosts.allow (exceptions)
#
# Public services
sendmail: ALL
in.ftpd: ALL
sshd: ALL

# Private services
in.fingerd: .mydomain.country LOCAL
in.cfingerd: .mydomain.country LOCAL
cfd: .mydomain.country LOCAL
sshd forwarded: .mydomain.country LOCAL

# Portmapper has to use IP series
portmap: 128.39.89. 128.39.74. 128.39.75.
```

and

```
# /etc/hosts.deny (default)
ALL: ALL
```

may be distributed to each host by cfengine:

```
copy:
/masterfiles/hosts.deny dest=/etc/hosts.deny
  mode=644
  server=trusted
/masterfiles/hosts.allow dest=/etc/hosts.allow
  mode=644
  server=trusted
```

and installed as follows:

```
editfiles:
{ /etc/inet/inetd.conf
  # Make sure we're using tcp wrappers
  ReplaceAll '/usr/sbin/in.ftpd' With '/local/sbin/tcpd'
  ReplaceAll '/usr/sbin/in.telnetd' With '/local/sbin/tcpd'
  ReplaceAll '/usr/sbin/in.rshd' With '/local/sbin/tcpd'
```
ReplaceAll "/usr/sbin/in.rlogind" With "/local/sbin/tcpd"
processes:
  "inetd" signal=hup

The services that we do not need should be removed altogether; there's no sense in tempting fate:

editfiles:
  
  /etc/inetd.conf
  # Eliminate unwanted services
  HashCommentLinesContaining "rwall"
  HashCommentLinesContaining "/usr/sbin/in.fingerd"
  HashCommentLinesContaining "comsat"
  HashCommentLinesContaining "exec"
  HashCommentLinesContaining "talk"
  HashCommentLinesContaining "echo"
  HashCommentLinesContaining "discard"
  HashCommentLinesContaining "charge"
  HashCommentLinesContaining "quotas"
  HashCommentLinesContaining "users"
  HashCommentLinesContaining "spray"
  HashCommentLinesContaining "sadmin"
  HashCommentLinesContaining "rstat"
  HashCommentLinesContaining "kcm"s
  HashCommentLinesContaining "coomsat"
  HashCommentLinesContaining "xaudio"
  HashCommentLinesContaining "uucp"
}

Process Monitoring
When it comes to process management, we are usually interested in three things:

- making sure certain processes are running
- making sure some processes are not running
- sending HUP signals to force configuration updates

To HUP a daemon and make sure that it is running, we write:

processes:
  linux::
    "inetd" signal=hup restart "/usr/sbin/inetd" useshell=false
    "xftp" restart "/local/sbin/xftp" useshell=false

The useshell option tells cfengine that it should not use a shell to start the program.
The idea here is to protect against IFS attacks. Unfortunately, some programs require a
shell in order to be started, but most do not. This is an extra precaution. When the cron
daemon crashes, restarting it can be a problem since it does not close its file descriptors properly when forking. The dumb option helps here:

"cron" matches=>1 restart "/etc/init.d/cron start" useshell=dumb

To kill processes that should not be running, we write:

processes:
  solaris::
    #
    # Don't want CDE stuff or SNMP peepholes...
    #
    "ttddserverd" signal=kil
Almost all security programs available monitor file integrity. Cfengine also incorporates tools for monitoring files.

```bash
'snmpd' signal=kill
'mibisa' signal=kill
A couple of years ago, a broken cracked account was revealed at Oslo College by the following test in the cfengine configuration:
processes:
    # Ping attack?
    'ping' signal=kill inform=true
There are few legitimate reasons to run the ping command more than a few times. The chance of cfengine detecting single pings is quite small. But coordinated ping attacks are another story. When it was revealed that a user had twenty ping processes attempting to send large ping packets to hosts in the United States, it was obvious that the account had been compromised. Fortunately for the recipient, the ping command was incorrectly phrased and would probably not have been noticed.
processes:
    'sshd'
    restart '/local/sbin/sshd'
    useshell=false
    'snmp' signal=kill
    'mibisa' signal=kill
    'named' matches=>l
    restart '/local/bind/bin/named'
    useshell=false
    # Do the network community a service and run this
    'identd' restart '/local/sbin/identd' inform=true
Process management also includes garbage collection, which we shall turn to later.

Monitoring Files
Almost all security programs available monitor file integrity. Cfengine also incorporates tools for monitoring files. Here are some of the elements in the fairly complex files command:
files:

classes::
/file-object
    mode=mode
    owner=uid-list
    group=gid-list
    action=fixall/warnall..
    ignore=pattern
    include=pattern
    exclude=pattern
    checksum=md5
    syslog=true/on/false/off
In addition to these, there are extra flags for BSD filesystems and ways of managing file ACLs for systems like NT. Here are some examples of basic checks on file permissions:

classes:
    have_shadow = ( `/bin/test -f /etc/shadow` )
NFServers = ( server1 server2 )
files:
    any::
        /etc/passwd mode=0644 o=root g=other action=fixplain
have_shadow:
    /etc/shadow mode=0400 o=root g=other action=fixplain
# Takes a while so do this at midnight and only on servers
NFSservers.Hr00::
    /usr/local
    mode=0002 # Check no files are writeable!
    recurse=inf
    owner=root,bin
    group=0,1,2,3,4,5,6,7,staff
    action=fixall

In the last example we parse through a whole file system (recurse=inf) and as a result get a number of checks for free. Any previously unknown setuid programs are reported, as well as any suspicious filenames. Let's elaborate.

The Setuid Log

Cfengine is always on the lookout for files that are setuid or setgid root. It doesn't go actively looking for them uninvited, but whenever you get cfengine to check a file or directory with the files feature, it will make a note of setuid programs it finds there. These are recorded in the file cfengine.host.log, which is stored under /etc/cfengine or /var/log/cfengine. When new setuid programs are discovered, a warning is printed, but only if you are root. If you ever want a complete list, delete the log file, and cfengine will think that all of the setuid programs it finds are new. The log file is not readable by normal users.

Suspicious Filenames

Whenever cfengine opens a directory and scans through files and directories (recursively: files, tidy, copy), it is also on the lookout for suspicious filenames, for example files like "../" containing only space and/or dots. Such files are seldom created by sensible sources, but are often used by crackers to try to hide dangerous programs. Cfengine warns about such files. Although not necessarily a security issue, cfengine can also warn about filenames that contain nonprintable characters and directories that are made to look like plain files by giving them filename extensions:

```
control:
# Security checks
#
NonAlphaNumFiles = ( on )
FileExtensions = ( o a c gif jpg html ) # etc
SuspiciousNames = ( .mo lrk3 lkr3 )
```

The file-extension list may be used to detect concealed directories during these searches and warn if users create directories that look like common files. Additional suspicious filenames can be checked for automatically.

The mail spool directory is a common place for users to try to hide downloaded files. These options inform about files that do not have the name of a user or are not owned by a valid user:

```
control:
    WarnNonOwnerMail = ( true )
    WarnNonUserMail = ( true ) # Warn about mail which is not owned by a user
```
The spirit of cfengine is not to bother us with warnings, but rather to fix things automatically.

Corresponding commands exist to delete these files without further ado. This can be a useful way of cleaning up after users whose accounts have been removed.

**Checksums and Tripwire Functionality**

Cfengine can be used to check for changes in files that only something as exacting as an MD5 checksum/digest can detect. If you specify a checksum database and activate checksum verification,

```plaintext
control:
    ChecksumDatabase = ( /etc/cfengine/cache.db )
    ChecksumUpdates = ( false )
files:
    /filename checksum=md5 ....
    /dirname checksum=md5 recurse=inf....
    # If the database isn't secure, nothing is secure...
    /etc/cfengine/cache.db mode=600 owner=root fixall
```

then cfengine will build a database of file checksums and warn you when files' checksums change. This makes cfengine act like Tripwire (currently only with MD5 checksums). It can be used to show up Trojan-horse versions of programs. It should be used sparingly, though, since database management and MD5 checksum computation are resource-intensive operations that could add significant time to a cfengine run. The `ChecksumUpdates` variable (normally false) can be set to true to update the checksum database when programs change for valid reasons.

Warnings are all well and good, but the spirit of cfengine is not to bother us with warnings, but rather to fix things automatically. If you are worried about the integrity of the system, then don't just warn about checksum mismatches here; make an md5 copy comparison against a read-only medium that has a correct, trusted version of the file on it. That way if a binary is compromised you will not only warn about it but also repair the damage immediately!

The control variable `ChecksumUpdates` may be switched to on in order to force cfengine to update its checksum database after warning of a change.

**FileExtensions**

This list may be used to define a number of extensions that are regarded as plain files by the system. As part of general security checking, cfengine will warn about any directories having names that use these extensions (possibly in order to conceal them):

```plaintext
FileExtensions = ( .c .o .gif .jpg .html )
```

**NonAlphaNumFiles**

If enabled, this option causes cfengine to detect and disable files that have purely non-alphanumeric filenames, that is, files that might be accidentally or deliberately concealed. The files are then marked with a suffix `.cf-nonalpha` and are rendered visible:

```plaintext
NonAlphaNumFiles = ( on )
```

These files can then be tidied (deleted) or disabled by searching for the suffix pattern. Note that nonalphanumeric means ASCII codes lower than 32 and higher than 126.

**Defensive Garbage Collection**

We tend to be worried about the fact that crackers will destroy our systems and make them unusable, but many operating systems are programmed to do this to themselves!
Few systems can survive a full system disk, yet many logging agents go on filling up disks without ever checking to see how full they are getting. In short, they choke themselves in a self-styled denial-of-service attack. Cfengine can help here by rotating logs frequently and by tidying temporary file directories:

disable:
  Tuesday.Hr00::
  #
  # Disabling these log files weekly prevents them from
  # growing so enormous that they fill the disk!
  #
  /local/iu/httpd/logs/access_log rotate=2
  /local/iu/httpd/logs/agent_log rotate=2
  /local/iu/httpd/logs/error_log rotate=2
  /local/iu/httpd/logs/referer_log rotate=2
  FTPserver.Sunday::
  /local/iu/logs/xferlog rotate=3
  tidy:
  /tmp pattern=* age=1

Process garbage collection is just as important. There are lots of reasons why process tables fill up with unterminated processes. One example is faulty X terminal software that does not kill its children at logout. Another is that programs like Netscape and pine tend to go into loops from which they never return, gradually loading the system with an ever-increasing glacial burden. If the host concerned has important duties, this lack of responsiveness can compromise key services. It also gives local users a way of carrying out denial-of-service attacks on the system. Just killing old processes can cause your system to spring back from its ice-age blues (hopefully without littering the system with too many dead mammoths or bronze-age ax bearers). If the host concerned has important duties, then this lack of responsiveness can compromise key services. It also gives local users a way of carrying out denial-of-service attacks on the system.

If users always log out at the end of the day and log in again the next day, then this is easy to address with cfengine. Here is some code to kill commonly hanging processes. Note that on BSD-like systems process options 'aux' are required to see the relevant processes:

processes:
  linux\freebsd\sun4::
    SetOptionString "aux"
  any::
    *Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec*
      signal=kill
      include=ftpd
      include=tcsd
      include=xterm
      include=netscape
      include=ftp
      include=pine
      include=perl
      include=irc
      include=java
      include=/bin/ls
      include=emacs
      include=passwd

If the host concerned has important duties, lack of responsiveness can compromise key services. It also gives local users a way of carrying out denial-of-service attacks on the system.
This pattern works like this: As processes become more than a day old, the name of the month appears in the date of the process start time. These are matched by the regular expression. The include lines then filter the list of the processes further, picking out lines that include the specified strings. On some BSD-like systems the default `ps` option string is `-ax` and you might need to reset it to something that adds the start date in order to make this work.

Another job for process management is to clean up processes that have hung, gone amok, or are left over from old logins. Here is a regular expression that detects non-root processes that have clocked more than 100 hours of CPU time. This is a depressingly common phenomenon when a program goes into an infinite loop. It can starve other processes of resources in a very efficient denial-of-service attack.

```
any::
  
  # Kill processes which have run on for too long e.g., 999:99 cpu
  # time
  # Careful a pattern to match 99:99 will kill everything!
  #
  *[0-9][0-9][0-9][0-9]:[0-9][0-9]* signal=term exclude=root
  *[0-9][0-9][0-9][0-9]:[0-9][0-9]* signal=term exclude=root
```

Under NT this is not so simple, since the process table for the cygwin library applies only to processes that have been started by programs working under the UNIX process emulation. Hopefully this shortcoming can be worked around at some point in the future.

The next and final thrilling installment discusses how to secure a Web and ftp site and details some of the ways that cfengine can protect you from attacks.
toolman

Revision Control Revisited

You’ve probably seen articles extolling the virtues of revision-control systems. I’m not about to argue with their premise, nor am I going to repeat the discussion. What I will talk about here is a method of automating what is probably the most common sequence of actions associated with revision control: the cycle of check-out, edit, check-in. Presumably, if this process is made easier, use of revision control in appropriate situations would become more likely; we’ll assume that this is a good thing. Also, if this process is more automated and reliable, then errors are presumably less likely (for example, forgetting to check files back in, or editing as root); this is also a good thing.

Background

In the UNIX world, the two most common packages used for revision control are SCCS (Source Code Control System) and RCS (Revision Control System). I like the former a little better, because I perceive it as being a bit easier to use (probably just a case of familiarity). But I’ve heard it said that the latter is more powerful, though that doesn’t matter much to me since my usage pattern is very simple: I’m just interested in checking out a single file at a time, editing it, and checking it back in. [1] Also, SCCS is subject to licensing restrictions, though it is bundled with many UNIXes, while RCS is freely available and is also the basis for CVS (Concurrent Versions System). Various other implementations of revision-control systems are also available, both commercially and otherwise, and with varying degrees of application specificity; see the Resources section below.

Typical Command Sequences

The most common sequence of actions used with revision-control systems (for me, at least) is: check-out, edit, check-in. With SCCS, this will often be done with commands like those below (italics indicates user keystrokes):

\% sccs edit filename
\% vi filename
\% sccs delget filename

Some variations are possible. Also, your editor religion, er, choice may vary.

With RCS, commands such as these might typically be used:

\% co -l filename
\% vi filename
\% ci -u filename

Again, variations are possible, even likely. Please see the SCCS and RCS man pages for details and subtleties on command syntax, usage, and options.

The above commands look easy (and they are), but they’re often a lot of typing for a few little changes, and errors are possible. To put it another way, when you have to use them over and over again, they’re tedious and a pain in the butt.

The Tool

To make the check-out, edit, check-in cycle a little more bearable, I composed a tool some years ago to automate this process, and gradually refined it and added various conveniences, sanity checks, and features as needed over time. For instance, it will first tell you if anything in the directory is already checked-out and will warn you if it detects that you are root-ly (as it is generally the custom not to intentionally modify

by Daniel E. Singer

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

<des@cs.duke.edu>
There are several opportunities to bail out, and you can even have it "unedit" the file.

revision-controlled files as root). Also, I first wrote it for just SCCS (recall my bias), but later added RCS capability for a colleague who preferred that.

Here's a screen listing of a run of the script:

% rc usenix-article <RETURN>
rc: Using editor "vi"...
rc: sccs check
rc: Nothing being edited.
Continue (y/n)? [y] <RETURN>
rc: Checking out "usenix-article"...
rc: sccs edit "usenix-article"
1.7
new delta 1.8
679 lines
rc: Edit "usenix-article" (y/n)? [y] <RETURN>
[... edit session here ...]
rc: Check in "usenix-article" (y/n)? [y] <RETURN>
rc: Checking in "usenix-article"...

*** COMMENTS ***
rc: sccs delget "usenix-article"
comments? revised the part about revisions <RETURN>
1.8
27 inserted
27 deleted
652 unchanged
1.8
679 lines
rc: Done.
%

Notice that other than typing the single command (with the very short command name), editing, adding the usual comment, and hitting the return key a few times, there was nothing else to type. There are several opportunities to bail out (or just type Control-C), and you can even have it "unedit" (in SCCS lingo) the file. Here's an example of that:

% rc usenix-article <RETURN>
rc: Using editor "vi"...
rc: sccs check
rc: Nothing being edited.
Continue (y/n)? [y] <RETURN>
rc: Checking out "usenix-article"...
rc: sccs edit "usenix-article"
1.8
new delta 1.9
679 lines
rc: Edit "usenix-article" (y/n)? [y] <RETURN>
[... edit session here ...]
rc: Check in "usenix-article" (y/n)? [y] n
rc: Unedit "usenix-article" (y/n)? [n] y
rc: Unediting "usenix-article"...
rc: sccs unedit "usenix-article"
usenix-article: removed
1.8
679 lines
rc: Done.
%
If the file you want to check out is already checked-out, \texttt{rc} (Revision Control) will warn you and ask if you want to continue. If you do continue, it will skip the check-out and go straight to the edit step.

If the file does not yet exist, \texttt{rc} will warn you and ask if you want to continue. If you do continue, it again will go straight to the edit step, and will then ask if you want to create a history file (that is, check it in after editing is done).

**Preference**

As mentioned, \texttt{rc} will work with both SCCS and RCS. In fact, it will do its pointy-headed best to figure out which to use in a given situation, using a more or less heuristic approach; that is, it takes its best guess.

In an ambiguous situation, or when the file does not already exist, \texttt{rc} will employ a preference that you can set either with a variable or a command-line option. The variable \texttt{RC\_FAVOR} can be set in the script itself or in the environment to be one of \texttt{SCCS} or \texttt{RCS}. This can be overridden by supplying either the \texttt{-s} or \texttt{-r} command-line option.

**Executabits**

One annoyance of SCCS (RCS is immune to this one) is the nonretention of the executable bits in the file permissions. That is, when you check a file out with SCCS, the working copy loses any execute bits that were set, and when you check the file back in, they stay lost. This is inconvenient, say, if your file is a script, and the working copy happens to also be the actual copy that gets used in real life.\cite{2} So, if for instance you have a file \texttt{welcome-to-our-site.cgi}, and you do a check-out, edit, check-in, and you forget to do the pesky \texttt{chmod} to reenable the execute bits, then your whole Web site is hosed, and you just lost your job, and you can no longer feed your family, and ... \texttt{rc} will track and set the execute bits both during the file check-out and later during the file check-in. Your family is safe!

**Sleight-of-Hand**

Since the editor used by \texttt{rc} can be determined by an environment variable, it's possible to have some action performed other than an interactive edit session.\cite{3} For instance, another script or program could instead be invoked to perform some automated update (and possibly also call up an interactive editor), with \texttt{rc} again serving as a wrapper to deal with the revision control before and after. (This is why I added the "Using editor" message at the beginning of the script output.) An outer wrapper script might be used to accomplish this; for example:

\begin{verbatim}
#!/bin/sh
VISUAL='wizbang -x' rc $*
\end{verbatim}

The next section describes an alternative approach to this.

**Embeddability**

I recently added some options to \texttt{rc} to make it more amenable to being used from within other scripts. I've been working on some code to implement a simple but flexible database system all via shell\cite{4}, and ran into an actual situation in which the database file that I wanted to somewhat transparently update was under SCCS. I wanted to incorporate some of the sanity checks and SCCS/RCS flexibility from \texttt{rc} but didn't want to try to add in all of that code. So I added to \texttt{rc} a "quiet" option and also options to isolate the check-out and check-in portions as separate invocations. After handling the updates in memory, the database script calls \texttt{rc -q dbfile} to check out the file, then updates the file, then calls \texttt{rc -i -c "some comment" dbfile} to check it back in.
rc is just plain simpler to use than SCCS or RCS directly, and this is especially true for naive users.

Thus, rc can not only be used as an interactive, command-line tool, but can also be embedded in other applications.

Pathnames
Another goodie provided by rc is the ability to specify your file with a full pathname. For example:

```
% rc ~/public_html/recipes/guacamole.html
```

No need to cd to the directory, modify the file, and then cd back, in those situations where you only need to adjust the coriander.

Simplicity
rc is just plain simpler to use than SCCS or RCS directly, and this is especially true for naive users. Let’s say you’ve got a secretary who needs to occasionally maintain some files that are under revision control. He doesn’t really need to be bothered with the intricacies of revision-control systems. You can just tell him, “Look, when you need to edit one of these files, just use this single, simple command. . . .”

Summary
So there you have it, a tool you can use to automate common usages of SCCS and RCS, reduce errors, and make life easier for both naive and sophisticated users.

Resources
On UNIX systems, see the online reference manual (the man pages) for sccs and rcs, to get all of the options and subcommands that are supported on your system.

RCS and CVS are available through the Free Software Foundation. See these Web pages for more information:

<http://www.fsf.org/software/rcs/>
<http://www.fsf.org/software/cvs/>  

The book on SCCS and RCS:


A good introductory article on RCS, and a pair on CVS:

An article on how to apply this stuff more directly to our trade:

A good place to find information about several revision-control systems is the Cyclic Software Web site, <http://www.cyclic.com/gallery/index.html>.

Another is this page at Stokely Consulting:
<http://www.stokely.com/unix.sysadm.resources/vrscrtl.html>
The `rc` Bourne shell script and associated documentation can be found at:

<http://www.cs.duke.edu/~des/toolman/>

`rcE` (RCs Edit), written by Rune Mossige <runemo@telenor.no>, is another Bourne shell script that is similar to `rc`. It works only with RCS and will attempt to provide an appropriate keyword header for the types of any files created. You can get it, too, at the Toolman Web page.

If you have a script that works with revision-control systems, please let me know about it and I'll add a link on the Toolman Web page.

Notes

[1] My use of revision control tends to fall within the scope of modifying system-configuration files and Web pages. I'm not involved with any large-scale code-development projects or such, which might involve more complex requirements.

[2] We'll defer the discussion of whether or not this is a good practice, that is, working on the file in place. It probably isn't.

[3] A similar "trick" can be done via the "editor" used by the `edquota` program; see a future Toolman article.

[4] This may also be the subject of a future article, if they don't lock me up first.
how-to
Set Up a Time Server

This How-To describes how to set up a time server on your local network — on a UNIX or NT platform — and points at information for installing client software on various platforms. Details on what time service is about can be found at <http://www.eecis.udel.edu/~ntp>. Another site, particularly good for US military organizations, is <http://tycho.usno.navy.mil/ntp.html>. This How-To covers only a simple install; many advanced options are available, although they’re not needed at a large percentage of sites. Further information and details on configuring advanced options are available at <http://www.eecis.udel.edu/~ntp/ntp_spool/html/index.html>.

Network Time Protocol (NTP)
A number of NTP software packages are available. The most common is the server from Dave Mills at the University of Delaware. On <http://www.eecis.udel.edu/~ntp> is a point-and-download for both versions 3 and 4 of the protocol. Version 4, as of this writing, is “current” but “not yet complete.” Version 4 releases prior to 4.0.95 are not supported on NT. Version 3 is available for both platforms.

NTP itself is not affected by the Y2K issue. The software running the protocol operates strictly on seconds and parts of seconds and is unaware of what a “year” is. Other packages (including the operating system) that use NTP may or may not have problems, but those are not directly related to NTP, only to how they use the results.

What You Need
- a UNIX machine or NT machine, built (and properly secured)
- root or administrator privileges
- working Internet connectivity, TCP/IP installed
- an operating Domain Name Service (client is sufficient, so long as it can gain access to a server; otherwise you will need to enter IP addresses in your ntp.conf and/or /etc/hosts files)
- working tools, in your path:
  UNIX (all but gzip can be either vendor or GNU versions):
  C compiler
  awk
  make
  ed
  tr
  sh
  grep
  gzip (to uncompress distribution)

NT:
  Visual C++ version 5.0 or better
  Perl version 5
  InstallShieldSDK, but not the one in VC++5.0, as that one doesn’t work well enough for the ntp release
  some “unzip” program
Steps

1. Obtain server software.

Always get the newest release:

by http: <http://www.eecs.udel.edu/~ntp> get one of these:
   xntp3-5.93e.tar.gz
   xntp3-5.93e-export.tar.gz
   ntp-4.0.95.tar.gz (or whichever is the newest release)

by ftp: <ftp.udel.edu/pub/ntp>:
   xntp3-5.93e.tar.gz
   xntp3-5.93e-export.tar.gz
   testing/ntp-4.0.95.tar.gz

Whichever you selected will be referred to below as <sourcefile>.

2. Unpack server distribution.

Select external servers. Pick at least three that are topologically close. If you are going to act as a server to hundreds of clients (e.g. a university), you might consider stratum-1 servers, otherwise select stratum-2 or stratum-3 servers. You will need to contact the managers of the servers you select, out of courtesy. A list of potential servers is at <http://www.eecs.udel.edu/~mills/ntp/servers.html>.

In an appropriate source directory:

gzip -d <sourcefile>, which will now appear as <sourcefile> without the .gz extension

tar xf <sourcefile>

cd into the new directory created.


If you have an old version of the time daemon running, turn it off.

UNIX:

SVR4-based: ps -ef | grep ntp, then kill the process running, or
/etc/rc2.d/S74xntpd stop

BSD-based: ps ax | grep ntp, then kill the process running

NT:

Use the services control panel to turn it off.

3A. UNIX

./configure

make

make install

Create a file in /etc, /etc/inet, or wherever appropriate to your vendor's OS (see section below on startup) called ntp.conf. The location is referred to below as <location>.
Sample file (you need to replace the server entries with legitimate fully qualified domain names of your selected servers, and ensure that the driftfile location matches where you plan to put that file):

```
# server clock.university.edu
server ntpmachine.company.com
server timestamp.dept.gov
driftfile /etc/ntp.drift
```

chmod 640 /<location>/ntpd.conf (e.g., /etc/inet/ntpd.conf on Solaris 2)
touch /etc/ntp.drift
chmod 640 /etc/ntp.drift

3B. NT

read ntp\scripts\wininstall\distrib\readme.nt
run bldrel.bat
run ntp\scripts\wininstall\distrib\install.bat

edit the ntp.conf file just installed (ntp.conf by default)

Sample file (you need to replace the server entries with legitimate fully qualified domain names of your selected servers, and ensure that the driftfile location matches where you just installed that file):

```
# server clock.university.edu
server ntpmachine.company.com
server timestamp.dept.gov
driftfile ntp.drift
```

4. Start server.

4A. UNIX

Enable ntp daemon at boot.

SVR4-based system: Edit the /etc/rc2.d/S74ntpd or equivalent file, ensure that its idea of location of ntp.conf and xntpd match the actual locations where you have installed them. If you do not have such a file, create one with the contents in the BSD entry below, adjusted to match locations.

BSD-based system: Edit the /etc/rc.local (or site-specific run-control file where you start local daemons) and add:

```
# if [ -r /etc/ntp.conf -a -x /usr/local/bin/xntpd ]
then
/usr/local/bin/xntpd
fi
```

ensuring that you use locations that match where you installed those two files.

Then, either reboot or start the daemon by hand with command:

SVR4-based system: /etc/rc2.d/S74ntpd start
BSD-based system: /usr/local/bin/xntpd
4B. NT

Use the Services control panel to make NTP autostart. Then, either reboot or start it manually from the panel.

5. Obtain client software.

Many packages are available for various platforms to allow them to synchronize their internal clocks to your new server. Each package has its own specific set of installation instructions. Some locations to check:

<http://www.txdirect.net/users/sfisher/clock.html>
<http://www.eecis.udel.edu/~ntp/software.html>

On Solaris, assuming you didn’t delete the NTP package from the build “customize” screen:

```
  cp /etc/inet/ntp.client /etc/inet/ntp.conf
```

Then change the line for multicastclient to read instead

```
  server xxx.xxx.xxx.xxx
```

where the xxx sections are the IP address (or fully qualified domain name if you have name service running) of your server. Then either reboot the client or start the time service by running `/etc/rc2.d/S74ntpd start`. 


enough SNMP to be dangerous, part 3

This is a series of articles dedicated to teaching typical UNIX system-administrator types – people who can compile public-domain software and have some idea about TCP/IP – how to do UNIX-style hackery with SNMP. It is not an elegant and systematic approach to SNMP, but it should give you enough background to be dangerous.

In part 1 (login, December 1998), we discussed the basics of SNMP and SNMP tools; in part 2 (login, August 1999), we turned these into a little tool for resolving “my uptime is better than yours” arguments. In part 3, we move on to counters and less frivolous uses of SNMP.

We’ve wandered through the system group of MIB-II, but there are a bunch of other mandatory groups we can look at on any SNMP device. Of those, my favorites are interface status and statistics information. For each network interface on the device, you can find out handy facts like how many packets it has received, how many of them were bad, and how many it had to discard. These are the sorts of counters that fancy network-administration programs use in order to do elegant things. They can, of course, be used for much stupider purposes. Unfortunately, there is a catch.

Counters in SNMP are supposed to behave like the odometer in a car. They go up. When they get to the top, they start over at 0. They are not actually supposed to reset on reboot (or at any other time), except when otherwise specified. So what happens if you see numbers like the following pair?

- Input unicast packets: 1,000
- Input unicast packet errors: 500,000

Well, there are three obvious possibilities.

1. This interface is on a remarkably broken network.
2. This device has gotten terribly confused.
3. The input unicast packet counter has rolled over, but the error counter hasn’t; the right number is in fact 4,294,967,295 + 1,000

Nothing within SNMP will disambiguate these three cases. Furthermore, if the problem is that the device is confused, and the device is actually SNMP-compliant, there’s nothing you can do about it – there is no way to reset the counters. The official position on this is that the actual counter values aren’t meaningful; it’s the rate of change of the counter values that is meaningful. This is great if you’re running network-management software that sits around monitoring things continuously, but less than useful if you’re just poking at things once in a while.

Fortunately or unfortunately, compliance with this part of the SNMP spec is just as good as compliance with any other piece. The devices I’ve looked at fall into three classes. My favorite ones reset the counter on reboot, which completely confuses continuous network-management software, but works well for me. My second favorites are the ones that actually implement the spec; it’s somewhat annoying, but I understand the reasoning, and at least I know what to expect. The ones that truly annoy me are our Octel voicemail machines, which have counters that not only fail to reset – they also fail to roll over. This would be marginally less annoying if they counted up to the full value of an SNMP counter (2^{32} - 1), but instead they only count up to 65,535 (2^{16} - 1). As a result, most of the more interesting ones pegged their counters within a week or so.
In practice, what I do about these counter issues is ignore them. I’m not writing professional SNMP management tools; I’m doing quick-and-dirty network debugging. I have a program to calculate error rates, and then I look at them. If they’re insane, I apply obvious human-mediated debugging techniques to sort out the three possible reasons. (For instance, does the network work at all? If so, then there is not a 500% error rate. Does the device return equally bizarre information if queried otherwise? If so, then perhaps it’s confused, and rebooting it will improve the world.)

With that in mind, let’s keep working through MIB-II. We’ve pretty thoroughly mined the system; the remaining parts of MIB-II are:

- interfaces
- at
- ip
- icmp
- tcp
- udp
- egp
- transmission
- snmp

First, let’s decide to ignore some of these forever. “at” is the address translation group, which is officially deprecated because its meaning is entirely dependent on the protocols your device happens to be speaking. If TCP/IP is the device’s favorite protocol, the “at” group is basically an exceedingly annoying way of representing the arp cache.

The “transmission” group has data about the transmission media underlying the interfaces, if it has anything at all, which doesn’t happen all that often. You probably don’t care; I certainly don’t.

The “egp” and “icmp” groups have information about their respective protocols, if the device implements them. Once again, this is all very well, but they’re just not very interesting protocols for most purposes. The “snmp” group is one of the best examples, outside of particle physics, of the Heisenberg effect, whereby observing something changes the value observed. Of course if you send an SNMP get command to get the value of snmpInPkts, which is the number of SNMP packets received, you increment the counter. Aside from this minor and recondite pleasure, the snmp group doesn’t have many applications – if you track how many requests you make to a machine, you can see if anybody else is playing with SNMP, but tracking them down is a separate and thornier problem.

That leaves us with the apparently useful “interfaces,” “ip,” “tcp,” and “udp” groups. Here’s a walk through part of the interfaces group, to illustrate how SNMP tables work:

```
interfaces.ifNumber.0 = 2
interfaces.ifTable.ifEntry.ifIndex.1 = 1
interfaces.ifTable.ifEntry.ifIndex.2 = 2
interfaces.ifTable.ifEntry.ifDescr.1 = Silicon Graphics cc Ethernet controller
interfaces.ifTable.ifEntry.ifDescr.2 = Silicon Graphics lo Loopback interface
interfaces.ifTable.ifEntry.ifType.1 = ethernetCsmacd(6)
interfaces.ifTable.ifEntry.ifType.2 = softwareLoopback(24)
interfaces.ifTable.ifEntry.ifMtu.1 = 1500
interfaces.ifTable.ifEntry.ifMtu.2 = 8304
interfaces.ifTable.ifEntry.ifSpeed.1 = Gauge: 10000000
interfaces.ifTable.ifEntry.ifSpeed.2 = Gauge: 20000000
```
That's right, it has 15 interfaces, numbered 1 through 14, and 23. That's OK.

interaces.ifTable.ifEntry.ifPhysAddress.1 = 80:69:2:f6:ff
interaces.ifTable.ifEntry.ifPhysAddress.2 =
interaces.ifTable.ifEntry.ifAdminStatus.1 = up(1)
interaces.ifTable.ifEntry.ifAdminStatus.2 = up(1)
interaces.ifTable.ifEntry.ifOperStatus.1 = up(1)
interaces.ifTable.ifEntry.ifOperStatus.2 = up(1)
interaces.ifTable.ifEntry.ifInOctets.1 = 2081101543
interaces.ifTable.ifEntry.ifInOctets.2 = 31835092
interaces.ifTable.ifEntry.ifInUcastPkts.1 = 3224161
interaces.ifTable.ifEntry.ifInUcastPkts.2 = 500898
interaces.ifTable.ifEntry.ifInNUcastPkts.1 = 926910
interaces.ifTable.ifEntry.ifInNUcastPkts.2 = 0

interfaces.ifNumber is a familiar, single-instance variable that tells how many interfaces the machine has. You were probably assuming that we'd been using "0" because SNMP counts starting at 0. As you can see, this is false. Actually, if there's anything to count, it is not allowed to start below 1. (In most cases, it will start at 1, but trusting in anything is unwise with SNMP.) And it gets worse -- check this out:

interfaces.ifNumber.0 = 15
interaces.ifTable.ifEntry.ifDescr.1 = Serial0/0
interaces.ifTable.ifEntry.ifDescr.2 = Serial0/1
interaces.ifTable.ifEntry.ifDescr.3 = Serial0/2
interaces.ifTable.ifEntry.ifDescr.4 = Serial0/3
interaces.ifTable.ifEntry.ifDescr.5 = Serial0/4
interaces.ifTable.ifEntry.ifDescr.6 = Serial0/5
interaces.ifTable.ifEntry.ifDescr.7 = Serial0/6
interaces.ifTable.ifEntry.ifDescr.8 = Serial0/7
interaces.ifTable.ifEntry.ifDescr.9 = Ethernet1/0
interaces.ifTable.ifEntry.ifDescr.10 = Ethernet1/1
interaces.ifTable.ifEntry.ifDescr.11 = Ethernet1/2
interaces.ifTable.ifEntry.ifDescr.12 = Ethernet1/3
interaces.ifTable.ifEntry.ifDescr.13 = FastEthernet2/0
interaces.ifTable.ifEntry.ifDescr.14 = FastEthernet2/1
interaces.ifTable.ifEntry.ifDescr.23 = Serial0/7.110

That's right, it has 15 interfaces, numbered 1 through 14, and 23. That's OK. It's allowed to do that. Of course, if you're trying to loop through all the interfaces, this makes life unpleasant. Fortunately, SNMP allows you to do a "get next." A get next on interfaces.ifTable.ifEntry.ifDescr.0 (which, you will note, doesn't exist, and is guaranteed not to) returns interfaces.ifTable.ifEntry.ifDescr.1 and its value. If you have a handy indicator like interfaces.ifNumber, you can "get next" the appropriate number of times. Otherwise, you may just have to keep going until the next object is either an error or something in another part of the tree.

So here's a version of a program I've actually used to debug network problems:

```perl
#!/usr/bin/perl
#
# tcpnprobs
# Elizabeth D. Zwicky
# zwicky@sgi.com
# July 1998
use CGI qw(:all);
use SNMP;
# This turns on formatted printing of variables
$SNMP::use_sprint_value = 1;
$ORANGE_THRES = 5;
$RED_THRES = 20;
```
print header;
print start_html(-title=>"TCP/IP error rates", -bgcolor=>"ffffff");
print h1("TCP/IP error rates");
# Up to you to figure out how to get this set as a parameter;
# the elegant way is to write up a form, but you could always just
# hand-type it as part of the URL, as in
# http://yourhost/tcpipprobs?hostname=hosttocheck
$hostname = param('hostname');
if ($sess = new SNMP::Session(DestHost=>$hostname)){
    # First we pull a bunch of nice, straightforward single-instance
    # variables.
    $tcpout = $sess->get(['tcp.tcplOutSegs', '0']);
    $tcpretrans = $sess->get(['tcp.reTransSegs', '0']);
    $bin = $sess->get(['ip.ipInReceives', '0']);
    $ipinheader = $sess->get(['ip.ipInHdrErrors', '0']);
    $ipinaddr = $sess->get(['ip.ipInAddrErrors', '0']);
    $ipdiscard = $sess->get(['ip.ipInDiscards', '0']);
    print h3("$hostname");
    print p("TCP: $tcpout packets out, $tcpretrans (*,
        &percent($tcpretrans, $tcpout).
        * percent) TCP retransmission errors <br>");
    print p("IP: $bin packets received, $ipinheader (*,
        &percent($ipinheader, $bin).
        * percent) header errors, $ipinaddr (*,
        &percent($ipinaddr, $bin).
        * percent) address errors");
    # And then we walk off into manipulating tables and multiple
    # instances...
    # This is the number of interfaces on the machine
    $interfaces = $sess->get(['interfaces.ifNumber', '0']);
    print "<table border = 2>
    \n";
    print TR (
        th('&nbsp'),
        th('Input Packets'), th('Input Errors'), th('Input Discards'),
        th('&nbsp'),
        th('Output Packets'), th('Output Errors'), th('Output Discards'));
    # And now we loop
    foreach $index (0..($interfaces - 1)){
        $interface =
            $sess ->
            getNext(['interfaces.ifTable.ifEntry.ifIndex',
                '$index']);
        $descr =
            $sess ->
            getNext(['interfaces.ifTable.ifEntry.ifDescr',
                '$interface']);
        $admin =
            $sess ->
            getNext(['interfaces.ifTable.ifEntry.ifAdminStatus',
                '$interface']);
        # Now you can do your thing...
$oper =
    $sess->
    get(['interfaces.ifTable.ifEntry.ifOperStatus','
        "$interface"]);
$unknown =
    $sess->
    get(['interfaces.ifTable.ifEntry.ifInUnknownProtos','
        "$interface"]);
$input =
    $sess->
    get(['interfaces.ifTable.ifEntry.ifInNUcastPkts','
        "$interface"]);
$input +=
    $sess->
    get(['interfaces.ifTable.ifEntry.ifInNUcastPkts','
        "$interface"]);
$inputerrs =
    $sess->
    get(['interfaces.ifTable.ifEntry.ifInErrors','
        "$interface"]);
$inputdisc =
    $sess->
    get(['interfaces.ifTable.ifEntry.ifInDiscards','
        "$interface"]);
$output =
    $sess->
    get(['interfaces.ifTable.ifEntry.ifOutNUcastPkts','
        "$interface"]);
$output +=
    $sess->
    get(['interfaces.ifTable.ifEntry.ifOutNUcastPkts','
        "$interface"]);
$outputdisc =
    $sess->
    get(['interfaces.ifTable.ifEntry.ifOutDiscards','
        "$interface"]);
$outputerrs =
    $sess->
    get(['interfaces.ifTable.ifEntry.ifOutErrors','
        "$interface"]);

print TR (  
    td($descr), td($admin), td($oper),
    td('&nbsp'),
    td($input),
    td("$inputerrs (" . &percent($inputerrs, $input) . ")"),
    td("$inputdisc (" . &percent($inputdisc, $input) . ")"),
    td('&nbsp'),
    td($output),
    td("$outputerrs (" . &percent($outputerrs, $output) . ")"),
    td("$outputdisc (" . &percent($outputdisc, $output) . ")")
    );
)

print "</table>
";
}
else (  
    print p(b("Could not bind to $hostname: $!")));
)
print end_html;
sub ppercent {
    my($num) = $_[0];
    my($denom) = $_[1];
    if ($denom <= 0) {
        return 0;
    } else {
        my($percent) = ($num * 100) / $denom;
        if ($percent > $RED_THRES) {
            return sprintf("<font color=red>%3.2f%%</font>", $percent);
        } elsif ($percent > $ORANGE_THRES) {
            return sprintf("<font color=orange>%3.2f%%</font>", $percent);
        } else {
            return sprintf("%3.2f%%", $percent);
        }
    }
}

There are a few tricks here that we haven’t already discussed. The “ip” group gives me separate numbers for “UcastPkts” (unicast packets) and “NUcastPkts” (non-unicast packets, i.e., multicasts and broadcasts). For my purposes, this is irrelevant, so I add them together.

There’s also this unpleasant-looking result:

TCP: Wrong Type (should be Counter): NULL packets out, Wrong Type (should be Counter): NULL (0 percent) TCP retransmission errors
IP: Wrong Type (should be Counter): NULL packets received, (0 percent) header errors, Wrong Type (should be Counter): NULL (0 percent) address errors

That’s a UNIX machine that doesn’t keep these statistics in its kernel and therefore doesn’t feed them to its SNMP agent, running into a combination of beautiful error handling (the library’s) and completely laissez-faire error nonhandling (mine). You could make the output more beautiful, but you can’t get blood out of a stone, or TCP retransmission statistics out of a machine running IRIX 5.3’s default SNMP agent.

You may wonder how I picked the precise variables I show here. It’s clear even from the excerpts I’ve shown that these are not all the variables that are available to me. This is more or less pure empiricism; I started with a program that displayed pretty nearly everything and got rid of all the ones that I never actually needed, until the remaining information fit well on a page. Depending on your point of view, this is either science at its best or hackery at its worst.

Next: Exploring MIBs on your own; we discover device-specific MIBs.
A new tool was created to solve the existing NIS problems and address other common NIS problems. The tool also had to work in the new NIS environment, where all hosts would be NIS masters.

% ci -u group
% make group

The makefile, provided by the OS vendor, would distribute maps from /usr/etc/yp. As RCS was employed, the flat file used to build the NIS maps from and distribute from should be the last version checked in to RCS.

A New Tool
As a result of the project to define new standards, a new tool was created to solve the existing NIS problems and address other common NIS problems. The tool also had to work in the new NIS environment, where all hosts would be NIS masters. Following are the requirements for the new tool, ypmake:

- Ypmake, the program, had to work for the top-level NIS master, which would control the NIS maps, and also for the second-tier to n-tier masters that would receive maps. Since all hosts were to be NIS masters, we wanted the flexibility of ypmake working with any number of tiers.
- There should be no changes required to the program when new maps are introduced, old maps are removed, or machines are added/retired.
- Where applicable, syntax checking must be performed. Adding the ability to syntax-check a map should not require any modifications to ypmake.
- Building new maps or new types of maps should not require any modifications to ypmake.
- Ypmake had to deal with maps under RCS control and, possibly, maps that were autogenerated or available via some database. In the case of version-controlled maps, the most recent checked-in version had to be pushed.
- Ypmake had to deal with locking the passwd file to synchronize changes with rpc.yppasswd.
- A configuration file had to control all of the build process performed by ypmake. Ypmake must not be site-dependent.
- Under no circumstances must a run of ypmake destroy or corrupt an existing DBM map.
- Under no circumstances could a map be updated on a client during a run of ypmake. The host responsible for distribution of a map must synchronize map updates remotely with the client.
- An option must exist for execution of arbitrary programs after a map has been built.
- Ymake had to work on HP-UX 9.05 and 10.20, Solaris 2.5.1, 2.6, and 2.7, IRIX 6.2 and 6.5, and Digital UNIX 4.0.
- Ymake had to provide the ability to update any host in the NIS domain.

The first attempt at meeting these requirements was by modifying an existing makefile and augmenting it with features from GNU make. As more complexity was added to the makefile to satisfy the requirements, it became obvious that a separate program was needed to manage the task. Coding ypmake in Perl provided some advantages:

- Perl modules could be imported from any directory and dynamically loaded at runtime. This provided the means to extend ypmake for site-specific modules.
- Fewer calls had to be made to external programs such as makedbm and revnetgroup, since Perl could build DBM maps natively and handle the munging of any map.
Command-line options could be used. With make, variables could be defined on the command line but "command-line" options could not be passed.

Standard output and error output were far more robust.

Signals such as CTRL-C were handled properly. Thus, backing out and cleaning up was handled correctly.

The end result was a tool, designed for a specific need, that worked well enough to be used with other NIS configurations. One of the main benefits of ypmake was centralized administration of all NIS maps for all NIS hosts, regardless of whether or not custom modifications were needed to be made to a map. This was controlled through the configuration file, /etc/ypmake.conf. Because all hosts were NIS masters at the client site, a local customization was done by adding a second configuration file, /etc/ypmasters.conf, which defined the group a host belonged to and who its master was (i.e., which master updated its flat files). While ypmake does not need /etc/ypmasters.conf for a typical NIS master/slave/client configuration, the configuration file can be used with other variations on the NIS model given earlier.

Before providing examples of how ypmake can be configured to work in the typical NIS master/slave/client model and with the alternate models described earlier, let us take a brief overview of the configuration file. An understanding of the configuration file is vital to understanding how modification of this file is all that is necessary to adapt ypmake to your NIS environment and to the configurations to be discussed later.

Ypmake Configuration File

The default location of the configuration file is /etc/ypmake.conf. This can be modified with the -c option. The configuration file is divided into two logical sections, variable assignments and instructions for building NIS maps. Variable assignments are of the form VAR = VALUE. Variables are used by ypmake and any of its modules. (This includes custom modules written for a site.) A simple configuration file looks like:

```bash
# Variables
rcs-repository = /usr/etc/yp
directory = /usr/etc/yp
lock-dir = /var/run/yp
log proprio = /usr/etc/yp/log
# Where sendmail is located
BuildDBM::aliases::sendmail = /usr/lib/sendmail
# Max number of groups a user can be a member of
SyntaxCheckers::group::max-groups = 16
# Additional shells allowed in 'login-shell' field of password
# file in addition to those in /etc/shells
SyntaxCheckers::password::shells = /bin/false

# Build ethers map the same on all hosts
ethers buildmap::flat:builddbm::ethers:pushmap::ethers
# Build group map differently on NIS master. File is under RCS
# control so check out most recent revision before building.
group buildmap::rcs:builddbm::group:pushmap::yppush
```

The configuration syntax looks similar to that used by the Berkeley UNIX automounter. Variables alter defaults used in the program and provide information used when maps are built. By convention, variables with a :: are used by Perl modules. They are also used by site-specific modules.
Ypmake contains no “special” knowledge about how to build a map. It is controlled entirely through the configuration file. The second section contains the instructions needed to build maps for all hosts. Each map entry in this section is called a location list, with each entry in the location list called a location selection. Location selections are split into selectors \( \text{var} = \text{value} \) and options \( \text{var} = \text{value} \). Only one configuration file is needed for an environment regardless of how many domains exist. Should different actions need to be taken on different hosts, a host selector is available in the location list defining what action to take for the given host.

The options in a location selection instruct ypmake on how a map should be built. If a selector is not present in a location selection, it is considered the default location selection. In the example above, all location selections are defaults, since no selectors exist.

Briefly, the buildmap option instructs ypmake on how to construct the data for the NIS map that will be used to build the DBM files. The value flat indicates that ypmake should read a flat file, while rcs indicates ypmake should perform an RCS checkout of the last checked-in revision of the NIS map being built. A timestamp file is used to determine whether a map should be rebuilt. For flat files, the timestamp of this file is used. For rcs files, the timestamp file contains the last revision that was built. The builddbm option indicates which Perl module is used to build the DBM representation of the map, and the pushmap option indicates how the map should be distributed.

Ypmake with the Typical NIS Master/Slave/Client Model

In this environment, ypmake runs only on the NIS master. The NIS slave receives DBM map updates via yypush on the NIS master. The NIS client receives no updates. A sample section of the configuration file for such an environment would look like:

```plaintext
    group buildmap;=flat;builddbm;=group;pushmap;=yypush
    netgroup buildmap;=flat;builddbm;=netgroup;pushmap;=yypush
    passwd buildmap;=passwd;builddbm;=passwd;pushmap;=yypush
```

Using the snippet above, the instructions for the group map indicate that flat files should be used as the source and the group Perl module should be used to build the DBM files. The file-repository variable specifies the location of flat files. A value of rcs would cause ypmake to “check out” from RCS the last checked-in revision of the group map and use that as the source for generating the DBM files. The rcs-repository variable specifies the location of the RCS repository.

Except for the passwd map, the remaining maps are handled in a similar fashion. The passwd map is treated differently because rpc.yppasswd modifies the NIS passwd file. Because of this, a special buildmap handler was created to perform locking on the passwd file prior to using it for generating the DBM map. This ensures synchronization between ypmake and rpc.yppasswd.

Ypmake with the Multiple NIS Domains Sharing the Same Map Information

In this environment, ypmake runs on all NIS masters. To properly keep the masters in sync, it makes sense to employ locking during the distribution of a map. This ensures that while the DBM representation of a map is being built, the source is not being modified during the build.

In addition to control of map distribution by the master, it might also be desirable to provide a mechanism whereby a specific host or hosts in certain domains can be updated quickly. It would be undesirable to distribute maps from the master only to have to wait for a cron job to kick off to update the individual NIS domains. Ypmake provides a solution to this problem.
A sample section of the configuration file for this environment would look like:

```plaintext
group buildmap=flat;builddbm=ethers;pushmap=yppush \\
    host=mapmaster;buildmap=rts;pushmap=rdist
hosts buildmap=flat;builddbm=ethers;pushmap=yppush \\
    host=mapmaster;buildmap=rts;pushmap=rdist
password buildmap=flat;builddbm=passwd;pushmap=yppush \\
    host=mapmaster;buildmap=rts;pushmap=rdist
```

Because one of the NIS masters in this environment will be canonical for the NIS maps, ypmake will behave differently on this host from the other NIS masters. On this host, mapmaster, ypmake will need to transfer the flat files used to build the DBM files on the remaining NIS masters. It does this with the `pushmap=rdist` option. In the previous example, we used `yppush` to distribute maps to slaves. As the top-level NIS master will distribute to other NIS masters, `yppush` cannot be used. The default location selection, applicable for all hosts except mapmaster, uses `yppush` to distribute maps to the remaining hosts, which are slaves.

To enable synchronization of map updates between mapmaster and the other NIS masters, `rdist` uses the procmail lockfile program. An `rdist` special is used to copy the new flat file to /tmp on the remote server, lock the map in the build directory, copy the new flat file to the build directory, and then unlock the map.

The `rdist` module uses the file `/etc/ypmasters.conf` to determine which hosts to push to. The format of this file is:

```
[host]:[master]:[domain]
```

[host] is the name of the host running ypmake, [master] the name of the host that distributes to it, and [domain] is a logical name assigned to groups of machines. While it could be the NIS domain name of [host], it need not be.

For the example above, the contents would be:

```
master1:mapmaster:domain1
master2:mapmaster:domain2
```

NIS slaves do not appear in this file, since the NIS `ypservers` map controls which hosts are distributed to via `yppush`. NIS clients are also absent, as they do not receive updates. Only hosts distributed to via `rdist` appear in this file. The [domain] portion is used as an argument to the `yppush` command-line option. However, if ypmake is to provide the ability to update any host (via the `-l` and `-A` or `-m` switches) in the environment, NIS masters must appear in `/etc/ypmasters.conf`. For an update to propagate to a client or slave, ypmake must first run on the host mapmaster, then on the NIS master serving the domain of the client or slave. With the use of the `-A` and `-m` switch and `/etc/ypmasters.conf`, ypmake can achieve both of these steps using the following invocation of `ypmake` and the sample `/etc/ypmasters.conf` file:

```
% ypmake -l -A domain1 group
(etc/ypmasters.conf)
mapmaster:mapmaster:domain0
master1:mapmaster:domain1
```

This causes ypmake to perform an NIS update to all hosts in the NIS domain `domain1`. Updates cannot be propagated directly to a slave or client because of the use of `yppush` as the distribution mechanism, and because the hostnames of slaves and clients are not listed in `/etc/ypmasters.conf`. Ypmake accomplishes the update of the NIS domain `domain1` by searching `/etc/ypmasters.conf` for the domain `domain1` to determine who the NIS master is. Ypmake then performs a local build on mapmaster (local builds do

---

Updates cannot be propagated directly to a slave or client because of the use of `yppush` as the distribution mechanism, and because the hostnames of slaves and clients are not listed in `/etc/ypmasters.conf`.
not update the timestamp file), rdists the group file to master1, then performs an ssh onto master1 and reruns ypparse -l -A domain1 group. This second invocation of ypparse will perform a yppush forcing the update of the all slaves and clients in domain1.

On the master host mapmaster, the source used to build the DBM maps is under RCS control (buildmap=-rcs). Because of this, the most recent “checked in” version of the map will be used as the source. If someone is editing the group file, the flat file being edited will not be used as the source. Rather, the version in RCS will be used. This eliminates the problem of distributing partially edited maps.

**Ypmake with All Hosts as NIS Masters**

In this environment, ypmake runs on all hosts, since all hosts are NIS masters. While this allows fine-grained control over how NIS is architected, it does lead to administrative overhead. Nevertheless, it’s cool. Also, ypmake was originally designed to work in such an environment (though examples have already been given describing how ypmake can work with other NIS configurations).

The following map of this NIS configuration will be used in the discussion to follow:

```
master
  | +--- master1
  |     | +--- master1a
  |     |     | +--- master1b
  |     | +--- master2
  |     |     | +--- master2a
  |     |     |     | +--- master2b
```

The `/etc/ypmasters.conf` file looks like:

```
master:master:domain0
master1:master:domain1
master1a:master1:domain1
master1b:master1:domain1
master2:master:domain2
master2a:master2:domain2
master2b:master2:domain2
```

All hosts are listed in `/etc/ypmasters.conf` because all are NIS masters and because the `rdist pushmap` module uses this list for hosts to distribute to. Rdist is used rather than `yppush` because no slaves exist in this environment. Because each host is a master, for a NIS update to propagate to all hosts, the host `master` runs ypmake, which invokes an `rdist` of the flat files to master1 and master2. The hosts master1 and master2 then perform a ypmake at a time scheduled in cron to update master1a, master1b, master2a, and master2b. Finally, these four hosts invoke a `ypflush` from cron to update their NIS maps. Because updates occur automatically via cron, this sequence of steps is automated in the background.

At this number of layers, manually updating a host becomes tedious. Because of this, if a map change is needed to propagate to all hosts in domain2, the following would need to occur:

```
master% ypparse
master% ssh master1
```
master1% ypmake
master1% ssh master2a ypmake
master1% ssh master2b ypmake

The -l or -A and -m options, as utilized in the previous section, are also used here to force propagations to a set of machines in a domain or to any number of machines. We go into more detail here on what ypmake is doing to accomplish this.

With these options, the following one-line invocation of ypmake will accomplish the same as the previous five manual entries:

```
master% ypmake -l -A domain2 group       # Update group map for
Creating lockfile for group map ... done # all hosts in domain2
creating group ... done
syntax check of group ... done
building group DEM files ... done
updating local NIS group map ... done
distributing group ... done
building on remote hosts ... [+master2a +master2b -master2a -master2b] done
```

```
master% ypmake -l -m master2a group       # Update group map for
Creating lockfile for group map ... done # host master2a
creating group ... done
syntax check of group ... done
building group DEM files ... done
updating local NIS group map ... done
distributing group ... done
building on remote hosts ... [+master2a -master2a] done
```

The -l option causes a local update to occur. It is used only if the host running ypmake needs to be updated. If the host is a master that is responsible for distribution to other hosts in its domain, the -l option negates the push. Coupled with the -A <domain> and -m <host> options, this causes ypmake to reexecute itself on the masters of hosts that distribute to hosts in <domain> or to <host>.

The complete execution tree for the above two lines follows. While only the first line is entered manually, the remaining are run "behind the scenes" by ypmake.

```
master% ypmake -l -A domain2 group
(on master) ssh master2 ypmake -l -A domain2 group
(on master2) ypmake -l -A domain2 group
(on master2) ssh master2a ypmake -l -A domain2 group
(on master2a) ypmake -l -A domain2 group
(on master2) ssh master2b ypmake -l -A domain2 group
(on master2b) ypmake -l -A domain2 group
master% ypmake -l -m master2a
(on master) ssh master2 ypmake -l -m master2a group
(on master2) ypmake -l -m master2a group
(on master2) ssh master2a ypmake -l -m master2a group
(on master2a) ypmake -l -m master2a group
```

This stream of ssh invocations occurs regardless of the number of tiers in the NIS hierarchy. Ypmake uses the /etc/ypmasters.conf file to determine the hosts to distribute to. Regardless of how deep the number of levels in your NIS hierarchy, it determines the hosts it can distribute to and trusts these hosts to do the same until, eventually, the destination hosts are reached. As indicated in the expanded list, ypmake is run on all hosts. A separate version of ypmake is not needed depending on the role the hosts plays.
The verbosity of the output indicates the progress made as ypmake builds a map. Errors are displayed during the run. Where possible, a log file is created giving details of any errors that occur.

Note that when distributing to host master2a, the host master performed an ssh to master2, since it was the master for master2a. This information is contained in the configuration file /etc/ypmasters.conf. Moreover, ypmake, from reading this configuration file, knew that master could not distribute to master2a directly. However, as master2 could, and master could distribute to master2, ypmake performed an ssh to master2 to accomplish the job.

**Ypmake Execution Trail**

The general output of ypmake will look like the following:

```
Creating lockfile for [map] map ... done
  creating [map] ... done
  syntax check of [map] ... done
  building [map] DEM files ... done
  updating local NIS [map] map ... done
  distributing [map] ... done
```

The verbosity of the output indicates the progress made as ypmake builds a map. Errors are displayed during the run. Where possible, a log file is created giving details of any errors that occur.

Ypmake creates a lockfile via procmail’s lockfile or fcntl() for the map it is working on. Therefore, while many ypmake processes can be running at the same time, more than one ypmake process cannot build the same map. This locking is also synchronized when maps (i.e., flat files) are being distributed. In this case, procmail’s lockfile must be used because the rdistr special-executed after the map is transferred performs a lock on the remote file repository before transferring the new file from the temporary location to the repository. Following the lock, a build of the map begins.

The creating [map] output indicates that the flat-file representation of the map was built. The flat-file representation is created by the buildmap option in a location selection. Values currently supported are flat, passwd, and rcs. Additional values can be added via dynamically loadable Perl modules. Thus, flat files could be extracted from an Oracle database if a BuildMap::Oracle Perl module were created and the buildmap option was specified as buildmap::oracle. Flat files are created in a temporary directory under /tmp/ypmake/[pid] (where ypmake does all its work). Outside of the main ypmake code, almost all Perl code is dynamically loaded. This is the case for bundled modules and site-specific modules. A custom buildmap module is needed for the password map because it needs to be locked to disable updates via rpc.yppasswd.

The checkmap option specifies syntax checking. The value for the checkmap option contains the Perl module used to perform the syntax check of the map. As with the buildmap option, custom syntax checkers can be added via Perl modules. To augment the list included with ypmake, additional values can be added to the checkmap option, separated by commas. Thus, to perform the ypmake group check and a site-specific group check, the checkmap option would appear as checkmap::group,site-group.

The building [map] DBM files output indicates that DBM representations of flat files are being built. Ypmake goes to great lengths to ensure that sufficient disk space exists before it replaces the existing DBM files. It replaces the current files as follows:

1. Build DBM maps in /tmp/ypmake/[pid]
2. Move to /var/yp/[domain]/[map].tmp
3. Backup existing NIS DBM maps to /var/yp/[domain]/[map].orig
4. Rename `/var/yp/[domain]/[map].tmp` to `/var/yp/[domain]/[map]
5. Remove `/var/yp/[domain]/[map].orig`

Until step 3, yppmake can deal with error recovery properly. If the process succeeds with step 2, sufficient disk space exists to replace the map.

The updating local NIS [map] map output is for hosts that need a `ypxfr -f -h localhost [map]` to use the new map information immediately. Digital UNIX is an example of such a host. Nevertheless, it is run on all hosts.

**Signal Handling**

Yppmake catches the HUP, INT, TERM, and QUIT signals. At startup, it registers a subroutine to handle each of these signals. During program execution, cleanup subroutines are registered during specific areas of the code to deal with backout conditions should an interrupt occur. As an example, when the passwd buildmap module is executed, it registers a function to unlock the password file should an interrupt occur. The same signal-handling routines also remove temporary files created during execution.

**Future Directions**

Yppmake only supports NIS. Support for NIS+ would be a benefit.

Distributing a map to a host without building it is done with the -A or -m option. However, distributing the flat file does not follow the mechanism of forcing updates on clients by trickling down until the `sshd`/`ypmake` pair arrives at the final host with the -l option. Instead, yppmake copies the flat file directly to the master.

Support for `rsync` as a distribution mechanism would allow quicker updates (difficult, though, since it doesn't support the `rdist` "special" operation). If an API for `rsync` were available, such support would be possible.
fundamentals of troubleshooting TCP/IP

As a senior system and network administrator, I have frequently observed a lack of understanding of computer networks in general and TCP/IP in particular. The problem is most acute in help-desk and first-level desktop support. Understanding TCP/IP will enable help-desk and first-level support to isolate network problems more quickly. Correct and efficient problem identification reduces the duration of outages, thereby minimizing lost-productivity costs and improving customer satisfaction.

Consider the following scenario. Around 7:30 a.m. a user in accounting calls the help desk to report that the financial database is down. The user is very anxious to get access, because month-end reports are due to the CFO by 10:00 a.m. The help desk hangs up and sends a 911 page to the on-call database administrator to check the database server. This may be the right thing to do. But what if the network connection between the user’s PC and the server is broken? In that case, the DBA is annoyed (not as much as by a page four hours earlier, but still annoyed) and time has been wasted. The problem still isn’t fixed, so someone has to be paged to check the network. But there may not be one single person for this. There might be someone who deals with routers and someone else responsible for the PC network card and patch cable. How can a technician determine whether or not the problem is network-related?

The purpose of this article is to provide a basic framework for help-desk and first-level support to understand the basics of TCP/IP networks and give them a process to help identify the probable causes of problems. I’ll start with a description of end-node configuration and TCP/IP addressing. End-nodes are devices such as PCs, printers, and servers. For convenience, I will simply refer to PCs. Then I’ll outline a flowchart of general troubleshooting steps and common tools that are available on Microsoft Windows PCs (and many other operating systems).

TCP/IP Configuration

When a PC is attached to a TCP/IP network, several parameters must be set. First, every device needs a unique network address. IP addresses are 32 bits long and are partitioned into a network portion and a host portion. The subnet mask’s job is to identify the network portion explicitly. Finally, the PC needs the address of a default gateway or router. Whenever the PC cannot talk directly to the intended destination, it sends the data to a router. The router’s job is to forward packets toward the final destination.

IP addresses are frequently written in a form known as dotted-decimal or dotted-quad. The 32-bit IP address is broken into four groups of eight bits. In binary, eight bits can represent values between 0 and 255. Valid addresses fall in the range of 1 to 254. A group of all ones or 255 in decimal represents a broadcast address. For example, 192.168.30.255 means all nodes on the 192.168.30.0 network, assuming classful addressing is used.

Originally, the Internet protocol provided huge, large, and big networks referred to as class A, B, and C respectively. A class A network used the first octet (or eight bits) for the network number, leaving 24 bits of host. This provides for 16,777,214 host addresses. Class B and C networks used the first two and three octets respectively. This scheme works very well with dotted-decimal notation.

Unfortunately, classful addressing could not keep up with the Internet's tremendous growth. As a result, people have been forced to switch to classless addressing and rout-
ing, or CIDR. Instead of using fixed increments of 8, 16, and 24 bits for the network number, any value between 1 and 30 can be used. The bits assigned to the network portion must begin with the leftmost bit and be contiguous. For example, 255.255.192.0 is a valid subnet mask of length 18 bits. However, 255.255.193.0 and 255.255.208.0 are invalid because there are gaps in bits of the mask when written in binary form.

It is still common for people to refer to class A, B, and C networks. However, this is usually just a way of saying 8, 16, and 24 bits respectively. Actual classes are determined by the following rules: If the first bit is a zero, then it's a class A address; if the first bit is one and the second bit is zero, it's a class B address; if the first two bits are both one and the third bit is a zero, then it's a class C address. Table 1 shows the possible address ranges based on these rules.

<table>
<thead>
<tr>
<th>Class</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.0.0.0</td>
<td>126.0.0.0</td>
</tr>
<tr>
<td>B</td>
<td>128.0.0.0</td>
<td>191.255.0.0</td>
</tr>
<tr>
<td>C</td>
<td>192.0.0.0</td>
<td>223.255.255.0</td>
</tr>
</tbody>
</table>

Table 1. Traditional Network Classes

Classful addresses implied a subnet mask. Today, a subnet mask must be specifically configured. The role of the subnet mask is “neighbor determination.” Consider a PC with IP address 172.16.24.13 that wants to send a packet to 172.16.30.5. If the two PCs are on the same subnet, then the sender and receiver can communicate directly. Otherwise, the send must hand the packet off to a router, specified by the default gateway, for delivery to the final destination.

The IP address 172.16.24.13 falls into the range of class B networks. Using a 16-bit mask, both the sender and receiver are on the same subnet (i.e., 172.16.0.0). However, the sender is configured with a subnet mask of 255.255.255.0. The sender applies this mask to its IP address using a binary-AND operation to extract the “network portion.” It performs the same computation on the destination address. If the two results match, then the destination is a neighbor. However, in this case, 172.16.24.0 and 172.24.30.0 are different, so the sender must use its default gateway.

The final step is to resolve the IP address into a data link address. For those familiar with the OSI seven-layer model: IP operates at layer three, the network layer. The network layer is capable of routing traffic between networks. The data link layer, layer two, handles physical addressing. It determines which devices attached to a wire listen to a given transmission. Layer-two addresses are known as MAC or burned-in addresses. Every network adapter must have a unique 48-bit address assigned by the manufacturer.

IP addresses are translated into MAC (hardware) addresses for transmission over the local Ethernet or token-ring network. The mechanism by which the sender discovers the MAC address is called ARP (Address Resolution Protocol). If the destination is a neighbor, then a frame is created using the recipient's MAC address as the destination field. Otherwise, the router's MAC address is used. In either case, the IP packet header will include the recipient's IP address in the destination field.

Finally, the domain-name system (DNS) needs to be mentioned. Technically, DNS is not required for IP to work. However, in practice the Internet would grind to a halt without it. Human beings are simply better at remembering names than numbers. Can you imagine how much fun it would be to surf the Web if you had to remember addresses like 204.71.200.74, 131.106.3.253, and 209.249.27.175 instead of
database that maps host names and IP addresses. The downside is that applications
often fail when names cannot be resolved. Since most software is written to work with
either names or addresses, temporarily substituting IP addresses for host names can be
a useful debugging technique and also a workaround for DNS problems.

Troubleshooting Methodology
TCP/IP networks are based on connectionless datagrams. A datagram can be thought
of as a postcard and the network as the postal service. A postcard has sections for the
recipient’s address and a brief message. The sender drops the postcard in a nearby mail-
box, where it is picked up by the postal service. The postal service forwards the message
from office to office on its way to the final destination. Two pieces of mail may travel
different routes or arrive in a different order from that in which they were sent. If for
some reason the card cannot be delivered, the post office may stamp the reason on it
and return it to the sender. Sometimes the mail just disappears, possibly forever.

Networks behave like the postal service, only much faster. However, in networking there are more
options for discovering which messages are getting through and which are not. The flowchart can be
used to help identify the most likely cause of connectivity problems.

Before starting on the process below you should verify that the PC has the configuration settings dis-
cussed earlier. Depending upon your site, these parameters may be set manually or you may be using
DHCP (Dynamic Host Configuration Protocol). You should also verify that the TCP/IP software was cor-
rectly installed and initialized on the machine. You can do this by pinging either the PC’s IP address or
127.0.0.1. This loopback address is reserved to refer to the local machine.

The ping program is a standard utility for testing network reachability. The term “ping” is an acronym
for packet internetwork groper. It works by sending an “ICMP echo request” packet to the destination. If
the destination receives an echo request, it is supposed to transmit an echo reply to the sender. The
ping application usually displays status messages in response to each echo request. The message is usually
“Reply from w.x.y.z,” with a round-trip time and hop-count metric, or “Request timed out.”
Occasionally, you will see status messages indicating “destination unreachable” or “TTL expired in tran-
sit.” These are clues that a network link may down or that there may be a routing loop. The exact error
messages will vary depending on the implementation of ping on your computer.

Like many network applications, ping understands host names and IP addresses. If you try to ping a host
name that cannot be resolved using DNS (or some other lookup), the ping software will usually indicate an unknown host or bad IP address. This may be the result of a problem with the DNS server itself, or it may be that the network connection to the DNS server is down. If the name server responds to ping attempts, then nslookup can be used to verify that the DNS process is alive.

If the destination does not respond to ping attempts, the next step is to determine whether or not the destination is a neighbor. If it is on the same subnet, it may be a cable, hub, or switch problem (i.e., layers one and two of the OSI model) or the PC may be hung or powered off. If the destination is not local, then the problem may be related to the router. However, because the default gateway is also a neighbor, it may still be a hardware problem.

If the default gateway responds to pings, then the traceroute utility can be used to explore the data path between the PC and the destination. Traceroute, which is also known as tracert and trace depending on your operating system, makes use of the time-to-live field of the IP header. Each time that a router processes a packet it decrements the TTL field. When the field reaches zero the router sends back an ICMP message informing the sender that the packet expired in transit. Traceroute starts out with a TTL of one and increments the counter until the final destination responds or some maximum value is reached. Even if you do not have access to the router for viewing its configuration, you can discover the path that the packets are traveling and where the path ends or loops back on itself.

**Summary**

Troubleshooting network problems can be challenging. This article provides a framework to guide the reader through some basic diagnostic tests. Each test provides more information about what is working and what is not. Running the tests is straightforward. The skill lies in assembling the individual pieces into an accurate picture of the situation. Developing diagnostic skills requires practice.

The process described in this article provides several fundamentals. You can become more effective by understanding your local environment. If your organization uses managed hubs or switches, these devices may also have IP addresses. Use ping to test connectivity to these as well as to the default gateway. Also, most sites use an addressing scheme that makes router addresses predictable. For example, our routers and switches have addresses in the range of 1 to 30 starting with 1. Talk with the network administrators about diagnostic tests that are appropriate for your environment.

Remember that when people report a problem they tend either to state the problem in terms of the high-level task they are trying to perform or to claim that a specific component is broken. Our hypothetical user reported that the “financial database was down.” It is easy to accept the user’s statement as true and forward the call to the DBA. However, taking the time to determine what is still working will help to identify which problems are network-related and which problems need to go to the appropriate application group.
privacy in the real world

by Dan Geer

Dan Geer has been in distributed system management and security for enough years to have had his knowledge base go obsolete at least twice. He has slowly become certain that, despite wishful thinking to the contrary, we cannot rely on technology to trump policy. If we care about the outcome of what passes for democracy, we have to make policymakers pay attention.

EDITOR’S NOTE:

This speech was delivered on May 20, 1999, at the SmartCard Forum in Washington, D.C., in a debate setting.

By later in the day, the participants included:

- Steve Ellis (Intel)
- Phil Roettinger (Du))
- Mark Rotenberg (Epic)
- Bill Reinsch (head, BXA)
- Taher el Gamal (Kroll O’Gara)
- Stewart Baker (formerly NSA)
- Gilles Lisimache (Gemplas)
- Barry Smith (FBI)
- Dan Geer (CertCo)

It is timely that we speak of privacy in the real world. Privacy is the boundary condition between rights and privileges, a boundary evidently in dispute everywhere and forever. If privacy is "the right to be left alone – the most comprehensive of rights, and the right most valued by civilized men,"[1] then its sanctity serves as a barometer for our civilization.

Because privacy is definitionally “the state of being free from unsanctioned intrusion,”[2] it begs the question of by whose sanction intrusions may occur, if not merely by force.

As much as “civilization is the process of setting man free from men,”[3] privacy is its coin.

Privacy, beyond all other endowments, is the one more blessed to give than to receive, as, save for love and water, privacy is the only gift that can be exchanged across any of humankind’s divides.

Because privacy tolerates our differences with “ain’t nobody’s business but your own,” it creates us equal in ways nothing else can ever hope to do.

Since “philosophical and legal analysis has often identified privacy as a precondition for the development of a coherent self,”[4] one must conclude that it is a mortal peril to give up privacy.

As “privacy is the power to selectively reveal oneself to the world,”[5] in choosing what to reveal, however idiosyncratically, we demonstrate our liberty.

Yes, it is timely that we speak, that we speak plainly, that we in fact speak extremely, for “extremism in the defense of liberty is no vice.”[6]

It is said that the wonderful thing about a small town is that you know everyone, while the terrible thing about a small town is that they all know you. Indeed, a coherent if nostalgic argument for a “transparent society”[7] can be made, one where there are no secrets, where there is no privacy, where everyone knows everyone else’s business, where unsolved crime is very nearly impossible, where neither need nor triumph is invisible, a place where everything that is not self-incriminating is therefore public. Even were you able to craft the consensus that we all would each tell each other the contents of our hearts while leaving our cameras on at all times, I’m afraid that in such a utopian society you would soon find some were more equal than others. In short, I reject the one extreme, that of glass houses for us all.

I have come to the conclusion that in all things it is bigness that is the enemy, neither ideology nor biology nor theology but bigness. Big business, big government, big labor, big money, big crime, big media, big religion – it is their bigness alone that predisposes them to predatory behavior.

The two economists Adam Smith[8] and Ronald Coase[9] described the nature of our economic interactions – Smith with his millennial ideal of small producers trading amongst themselves in the mutual self-interest of wealth maximization, and Coase with his explanation of why the millennium does not arrive. In particular, Coase observed that economically viable firms expand until intra-firm coordination costs exceed inter-firm transaction costs. Putting it into biologic analogy, cells grow until their surface-to-volume ratio crosses a survivability threshold. Despite the starry enthusiasm of many
Internet devotees, it is now unarguably clear that although the Internet does lower transaction costs spectacularly, it lowers coordination costs more. Any reading of the newspaper will show you that the Internet is driving the greatest economic concentration in world history – the outscale prices of "Internet stocks" do not represent wealth creation, they represent wealth redistribution.

It is precisely this side effect of the global concentration of the control of wealth and economic power that must be the foundation of our thinking about privacy. As the ever-persuasive Phil Agre put it:

The global integration of the economy is . . . commonly held to decentralize political power by preventing governments from taking actions that can be reversed through cross-border arbitrage. But political power is becoming centralized in equally important ways: the power of national governments is not so much disappearing as shifting to a haphazard collection of undemocratic and nontransparent global treaty organizations, and the power to influence these organizations is likewise concentrating in the ever-fewer global firms. These observations are not pleasant or fashionable, but they are nonetheless true.[10]

If the reason I reject the transparent society is that I acknowledge my inability to sufficiently police its stronger members, then the most important thing I can do is to protect my privacy – and, frankly, at all costs. The loss of privacy is irreversible, for information is never un-revealed. Privacy is therefore the paragon of Hume's conjecture: Few liberties are lost all at once. In the face of the snowballing bigness of the institutions of globalized human life, we must reserve privacy rights explicitly so that we may misrepresent ourselves to those against whom we have no other defense, against those for whom our name is but a label on data collected without our consent.

Consider your own life. Perhaps there is indeed no one fact about you that you wouldn't good-naturedly share with this audience if I asked you politely. But by the time I got to twenty questions, few of you would still think this an amusing parlor game. The risk to you grows as the product of the number of personal facts times the number of potential recipients, but it is hard to fabricate an example where the benefit grows as fast even if you are a politician or otherwise live by publicity alone. On purely risk-management grounds, any finite tolerance for risk caps the amount of information you will want to play. This has nothing whatsoever to do with whether you have anything to hide. If for no other reason, we must make it understood that just as "there is nothing sinister in so arranging one's affairs as to [minimize] taxes,"[11] neither is there anything sinister in maximizing privacy. Naturally, the technologic tools of privacy can be misused, but what is it that is marvelous that cannot also be misapplied?

A wise man of my acquaintance, a career man in federal law enforcement, reacted to my arguments by telling me that I was typically naive. He said that my choice is not between Big Brother or no Big Brother; rather, it is between one Big Brother and lots of Little Brothers. He suggests that I think carefully before I choose.

I've thought about that a lot. I've thought about the comfort of being taken care of against the unease of having to be. I've compared the low cost of "one size fits all" to its correspondingly low benefit. I've thought hard about the proposition that the price of freedom is the possibility of crime. I've accepted that there is no such thing as righteousness if there is no possibility of sin. I conclude that privacy is worth its price, that near-absolute privacy is indeed the worst of all social constructs, except for all the others.
Our government is relentlessly pursuing an antiprivacy track that would not be so dangerous were it not so far outside the ken of the average person’s intuition or were we not the world’s presumed leader in matters of liberty.

Look around you. The price of duplicating electronic information is zero to begin with, and communications prices are dropping like a rock in hard vacuum. Ten years of progress in network computing has delivered on its original charge—location independence—to such a degree that location irrelevance is more like it. For the massless assets of an electronic world, jurisdictional legal boundaries are unutterably meaningless except where choice of law is prenegotiated. The signal-to-noise ROI of a commercialized Internet dismisses personal differences as an error condition to be corrected with the aggregated data of surveillance. Sans the Cold War, spooks everywhere are looking for commercial work, and technology drives policy through the obduracy of its artifacts—investments are sunk before democratic institutions detect their existence. Do you actually imagine that within such a dynamic you will be consistently able to count on your fellow man respecting your privacy or that you would have enforceable recourse against its diminution?

Governments everywhere hate privacy because the efficiency of regulation is proportional to the perfection of its surveillance. Here at home, our government is relentlessly pursuing an antiprivacy track that would not be so dangerous were it not so far outside the ken of the average person’s intuition or were we not the world’s presumed leader in matters of liberty. Beyond all other lessons, history teaches us that wherever personal boundaries are not taboo, the seeds of totalitarianism find fertile ground.

If only it were so simple that embattled farmers could again assemble by that rude bridge that arched the flood and fire the shot heard round the world.[12] The citizenry, ensnared to the demands of a culture of convenience, resplendent with glossy temptations to half the deadly sins and making entertainment of the rest, can hardly be counted upon to bite the hand that seems to feed them. The rate of change in what is within the realm of the technically possible is too great to digest, and we can oh so easily return to a world of sorcerers, alchemy, and faith in powers in proportion to their mystery.

When the cost of failure is intolerable, security designers insist that what is not explicitly permitted be forbidden. Because privacy is that thing whose loss is intolerable, we must make all acquisition and use of personal information forbidden absent explicit permission to do otherwise. We do that in the law itself—my attorney cannot act outside my permission, and my personal information is sacrosanct. We have to do that everywhere, or we have to let a globalized market take its course.

We already have many evidences of the market value of personal information. The affinity card at the grocery store pegs the market price of privacy there at about 5 percent. The cable-telephone provider who will take a $100 deposit in lieu of a credit check establishes a market price for that form of privacy. Many Web merchants measure their profit in customer data more than dollars. Anywhere the same price is charged for cash as for credit, the merchant’s credit-card discount rate is your premium for privacy. The foregone benefits of any frequent-buyer plan are what you’ll pay to avoid data fusion on your buying habits. The extra time you spend at the gas station is the price you pay for withholding information from their computers. The examples are legion, and many are far less prosaic than these. To leap to the other end of the spectrum, the European Union is considering extending privacy protections to legal persons, which they evaluate as an essential bulwark against becoming an electronic colony of ours.

The question before you is not preservation of the status quo—all hope of that is now lost. The question before you is whether a fervent unity is worth the effort. If by our
nonnegotiable demands we manage to harden the protections of privacy yet we are somehow ultimately shown to be wrong-headed, we have then merely to relax and enjoy it. If we fail to make privacy our hallmark and the effects are dire, we do not recover, for to do so is to unwind history.

Demand privacy, while the question is still relevant.

Notes
email archiving

Recently I was asked to log a copy of all of my company's incoming/outgoing email. Management thought it would be good if we could provide an email archive that users could search through if the documents and memos in our electronic library were inadequate. After searching the newsgroups I found a lot of people asking the same question, but no one had a satisfactory answer – until I came across a posting from Robert Harker that listed a sendmail feature that he called "copyuser." [1] This appeared to be what I needed. After installing it and doing some testing, I found that it was missing a few things. Harker's version logged all external messages, but it did not log any messages that were local. So I made some modifications to log a copy of all local and external mail to an account called "copyuser."

```
msgidruleset.m4
```

I'm not claiming to be an expert on writing sendmail rulesets. But with the recent thread on "Email Monitoring" going across the <sage-members> mailing list, I thought I'd share my solution. There may be a more efficient way to do this, but this seems to work for me. Listing 1 shows my modified version of Harker's configuration.

```
1 VERSIONID('msgidruleset.m4')
2 VERSIONID('Copyright 1998, Robert Harker, Harker Systems')
3 VERSIONID('www.harker.com, info@harker.com, 408-295-6239')
4 VERSIONID('Permission granted to use this as long as this')
5 VERSIONID('VERSIONID information is preserved in the M4 macro file')
6 VERSIONID('and any sendmail.cf files created using this M4 macro file')
7 VERSIONID('Modified to handle logging local mail as well as external mail')
8 VERSIONID('Shane B. Milburn, milburn@netcom.com, 04/19/1999')
9 ifndef (_MAILER_smtp_ ,
10 `errprint (`*** MAILER(smtp) must appear before copymail mailer`')`)dnl
11
12 LOCAL_CONFIG
13 CPNOCOPY
14
15 LOCAL_NET_CONFIG
16
17 LOCAL_RULE_0
18 $s+.NOCOPY.
19 $s+@mcst.gsfc.nasa.gov.NOCOPY.
20 $s+@$%.NOCOPY.
21 $s+@$%.copymail $snohostneeded $s1@$s2.NOCOPY
22 $s+@$%.copymail $snohostneeded $s1@$s2.NOCOPY
23 $s+ @copymail $snohostneeded $s1@$s2.NOCOPY
24
25 MAILER_DEFINITIONS
26 # Copy a message by sending it back to sendmail with an
27 # additional address: copyuser
28 copymail, P=/usr/lib/sendmail, P=mDFMuX, S=11/31, R=21, E=

L=990,
29 T=DNS/RFC822/X-Unix,
30 A=/usr/lib/sendmail copyuser@$s.NOCOPY $u
```
Here's a description of what msgidruleset.m4 does. The first eight lines will insert the text between the quotes into your sendmail.cf file as comments. Lines 25–30 define a new mailer named copymail. This definition reinvokes sendmail with the original recipients ($u) and an additional recipient named copyuser@$j.NOCOPY. While this does cause additional load on the server, in my case it was not enough of a load to cause concern.

Now let's look at the rest of the m4 file. Lines 9–10 will print an error message if in your site-config.mc file you try to declare - before smtp. The next two lines tell sendmail to declare a local class. The LOCAL_NET_CONFIG line forwards nonlocal network stuff to SMART_HOST. The LOCAL_RULE_0 statement is used to introduce new parsing rules. This is where you place any custom delivery agents and parsing rules you have defined. In Listing 1, the custom parsing rules are lines 18–23.

**Installation**

In order to install this, you need to place msgidruleset.m4 into a file called msgidruleset.m4 in the sendmail-8.9.3/cf/feature/ directory. Don’t forget that there are tabs between the first and second entries on lines 18–23. (If you do forget, sendmail will remind you.) Now add the following line to your site-config.mc file.

```plaintext
FEATURE(msgidruleset)
```

Here is what my site-config.mc file looks like.

```plaintext
VERSIONID("0(#)mcsst-config.mc Shane B. Milburn 04/21/1999")
VERSIONID("0(#)This configuration logs ALL email to copyuser.")
OSTYPE(solaris2)
FEATURE(use_cw_file)dnl
FEATURE(relay_entire_domain)dnl
FEATURE(always_add_domain)
FEATURE(crl)
MAILER(smtp)
MAILER(local)
FEATURE(msgidruleset)
```

After you create your site-config.mc file, use the m4 program to generate your sendmail.cf file. In /usr/local/src/sendmail-8.9.3/cf/cf you would use m4 ../m4/cf.m4 site-config.mc > sendmail.cf. This would create a sendmail.cf in the cf directory. You can either move this file into /etc/mail/ or invoke sendmail with the -C option to test. (Note: If you use the -C option for testing, make sure you change line 30 in Listing 1 to A=/usr/lib/sendmail -C/path/to/sendmail.cf copyuser@$j.NOCOPY $u. Otherwise, when you reinvoke sendmail it uses /etc/mail/sendmail.cf, which does not have the .NOCOPY parsing rules and will bounce your message.)

**Testing**

To test this new configuration, invoke sendmail from the command line. The first test takes a username mkephart and passes it to ruleset 3 and then 0. This causes the username to be rewritten to mkephart@mcsst.gsfc.nasa.gov.NOCOPY. Even though it's not shown, line 30 in msgidruleset.m4 also causes a second address copyuser@mcsst.gsfc.nasa.gov.NOCOPY to be passed back to sendmail.

```plaintext
# /usr/lib/sendmail -C/usr/local/src/sendmail-8.9.3/cf/cf/
sendmail.cf -bt > 3,0 mkephart
rewrite: ruleset 3 input: mkephart
rewrite: ruleset 96 input: mkephart
rewrite: ruleset 96 returns: mkephart
rewrite: ruleset 3 returns: mkephart
rewrite: ruleset 0 input: mkephart
```
Continuing with the testing, let's pass mkephart@mcast.gsfc.nasa.gov.NOOPCOPY to rulesets 3,0. You can see that this time the address is eventually passed to ruleset 98, which is lines 18–23. Notice that the .NOOPCOPY tag gets stripped off, and the email message is delivered to mkephart. Same as before, this is also happening for copyuser@mcast.gsfc.nasa.gov.NOOPCOPY and being delivered via mail.local to copyuser.

```
> 3,0 mkephart@mcast.gsfc.nasa.gov.NOOPCOPY
rewrite: ruleset 3 input: mkephart @ mcast . gsfc . nasa . gov . NOOPCOPY
rewrite: ruleset 96 input: mkephart @ mcast . gsfc . nasa . gov . NOOPCOPY.
rewrite: ruleset 96 returns: mkephart < @ mcast . gsfc . nasa . gov . NOOPCOPY .>
rewrite: ruleset 3 returns: mkephart < @ mcast . gsfc . nasa . gov . NOOPCOPY .>
rewrite: ruleset 0 input: mkephart < @ mcast . gsfc . nasa . gov . NOOPCOPY .>
rewrite: ruleset 199 input: mkephart < @ mcast . gsfc . nasa . gov . NOOPCOPY .>
rewrite: ruleset 199 returns: mkephart < @ mcast . gsfc . nasa . gov . NOOPCOPY .>
rewrite: ruleset 98 input: mkephart < @ mcast . gsfc . nasa . gov . NOOPCOPY .>
rewrite: ruleset 98 returns: mkephart < @ mcast . gsfc . nasa . gov . NOOPCOPY .>
```

Everything looks like it works, but let's try sending an email message and make sure it actually works before we put this sendmail.cf file into production.

```
# /usr/lib/sendmail -C/path/to/sendmail.cf -t
To: milburn@netcom.com
Subject: testing logging ALL email.
This is a test message. Did you receive a copy and is there a copy of this message in the "copyuser" mail folder?
  -shane
```

Checking my Netcom account, I see that I did receive the email. If I cat /var/mail/copyuser, it contains a copy of the message. Now that you're convinced it works, you can install your new sendmail.cf into /etc/mail/sendmail.cf and restart sendmail.

There is a catch to all of this logging. Depending on the amount of mail that goes through the system, /var/mail/copyuser can get quite large. Since I needed to make an archive through which a user must parse, it made sense to rotate the file daily. This allowed the user to grep a particular day's email rather than a week's or a month's worth of email. At the end of the week I staired the files into a weekending.MMDDYYYY.tar file and wrote it to 8mm tape. Before you implement an email archive, make sure your company has a policy about privacy issues and who actually owns the email sent to/from your server.

Reference

Other Resources

Newsread: <comp.mail.sendmail>
Config file downloads: <http://mcast.gsfc.nasa.gov/sendmail/>
the tclsh spot

Adding Hypertext Links and Image Display to htmlview.tcl

August's Tcsh Spot introduced an HTML viewer for older, curses-based email programs like elm and pine. By design, the viewer didn't support loading images or hypertext links. I usually don't want to waste time waiting for images to load while I'm reading my mail, and I seldom want to bounce to a hypertext link. But sometimes I do want that functionality, so...

This article will describe how to add hypertext links and image display to the htmlview.tcl package. Along the way, we'll briefly examine some of Tcl's support for large-scale projects (the namespace and package commands), binding events to actions, accessing the Web with the http package, and the image object.

You can find more details in my book, Tcl/Tk for Real Programmers; Steve Ball's Web Tcl Complete; and Mike Doyle and Hattie Schroeder's Interactive Web Applications in Tcl/Tk.

Two of the Tcl commands that simplify large-scale programming projects are the namespace and package commands. These commands support two software-engineering concepts that help construct easily maintained, modular code. The namespace command hides a set of commands and data in a private area where they won't interact with other code. The package command groups a set of procedures (possibly in several source files) into a single entity that can be accessed by name and optionally selected by revision number.

A namespace is similar in some respects to a C++ or Java class. Like a class, the items within a namespace may be either procedures or data. The data hidden within a namespace is available to all members of that namespace without being visible from the global scope (unless a script specifically requests the item). Tcl namespaces do not support inheritance, but namespaces can be nested.

Items within a namespace are named with a path-style naming convention, similar to the way X-11 windows or directory paths are named. The path separator for Tcl namespaces is the double colon (::). For example, an item baz within a namespace bar that's included within a namespace foo would be named as ::foo::bar::baz.

A namespace is created with the namespace eval command:

Syntax: namespace eval namespaceID arg ?args?

namespace eval Create a namespace, and evaluate the script arg in that scope. If more than one argument is present, the arguments are concatenated together into a single command to be evaluated.

namespaceID The identifying name for this namespace

arg ?args? The script or scripts to evaluate within namespace namespaceID

For example, this code will create a fastfood namespace, with internal variables for burgers and fries, and a burgerSeller procedure for modifying the number of burgers. By keeping the burger count inside the namespace, the customers can access a burger only by interacting with a burgerSeller.

namespace eval fastfood {
    variable burgers
    variable fries

```
proc burgerSeller {} {
    variable burgers
    incr burger -1
}

Selected components within a namespace can be made easily accessible with the namespace export and namespace import commands, or they can be accessed with the full namespace path identifier.

The http package hides all of its functions within the http namespace. Since the modified htmlview package will only use the http namespace, the simplest way to access the http functions will be via their full path. For example, the command::http::geturl evaluates the procedure geturl within the http namespace. Note that this doesn't invoke the procedure defined within the http namespace in the current namespace; it evaluates the geturl procedure within the http namespace.

The package command is a librarian, similar in some respects to the UNIX ar command. It joins a bunch of related procedures so that they can be accessed with a single name. Unlike the ar command, the package command doesn't create a new file for the joined data. The package create command creates a directory file (pkgIndex.tcl) that's used to determine which files should be loaded to resolve procedure references at runtime.

A Tcl script uses the package require command to declare that it will need to access a previously defined package. The script can also declare whether it requires a particular revision of this package, the latest revision, or the most recent minor revision of a particular major revision.

**Syntax:** package require ?-exact? packageName ?versionNum?

- `package require` Informns the Tcl interpreter that this package may be needed during the execution of this script. The Tcl interpreter will attempt to find the package and be prepared to load it when required.

- `?versionNum?` The name of the package to load.

- `-exact` If this flag is present, then versionNum must also be present. The Tcl interpreter will load only that version of the package.

So, with these two commands, we can look at how to use the http package. The two most-used commands in the http package are:

- `http::geturl url` download data from a URL and return a token to use to access this data in the future.

- `http::data token` return the data associated with a token.

This script downloads and prints the contents of the Scriptics homepage:

```tcl
package require http
set token [http::geturl www.scriptics.com]
puts [http::data $token]
```
The htmlib.tcl package has hooks to handle hypertext links. The htmlib.tcl package creates a binding on the text between an `<A>` and `</A>`. When a user clicks on that text, the procedure `HMLink_callback` will be invoked.

**Syntax:** `HMLink_callback win href`

- `HMLink_callback` A procedure to handle a hypertext link.
- `win` The currently active window.
- `href` The URL for the hypertext link.

Adding this code to the htmlib.tcl will enable the HTML viewer to access links:

```tcl
package require htmlib
proc HMLink_callback {win href} {
    # Clear the old contents from the window.
    HMRset_win $win
    # Get the url:
    set token [http::geturl $href]
    # Display the new text.
    HMParse_html $win "HMRender [http::data $token]"
}
```

That takes care of links. How about images? Since I still don’t want all images to be loaded when I read some spam by accident, I want to be able to click on a blank image and load just that image. (Cynical folks might notice that this selection mechanism lets me avoid banner ads.)

When the htmlib.tcl package sees an `<img ...>` tag, it evaluates the `HMMset_image` procedure with three arguments: the name of the primary text window, the name of the window that marks the location for this image, and the URL of the image. When the image has been loaded, `HMMset_image` will invoke `HMMgot_image` with the location marker and the new image data.

A normal browser would load the image data in `HMMset_image` and immediately invoke `HMMgot_image` to display it. Since we’d rather select the images we’re interested in seeing, we’ll bind an action to the marker label and load the image when the label is clicked.

The Tcl `bind` command lets a script define an event that can happen to a graphic object and define a script to evaluate when that event occurs.

**Syntax:** `bind widgetName eventType script`

- `bind` Define an action to be executed when an event associated with this widget occurs.
- `widgetName` The widget to have an action bound to it.
- `eventType` The event to trigger this action. Events can be defined in several formats:
  - `alphanumeric` A single printable (alphanumeric or punctuation) character defines a `KeyPress` event for that character.
  - `<modifier-type-detail>` This is similar to the X windows event descriptors that precisely define any event that can occur. Event descriptors like `<KeyPress-1>` are most common.
- `script` The script to evaluate when the event occurs while this window has focus.

Since I still don’t want all images to be loaded when I read some spam by accident, I want to be able to click on a blank image and load just that image. (Cynical folks might notice that this selection mechanism lets me avoid banner ads.)
If we define a new procedure `HMimage_request` to actually acquire and display the image, we can implement `HMset_image` like this to create a binding on the label that will invoke `HMimage_request` when the label is clicked. The `$src` variable contains the URL for the image. By calling `HMgot_image` with the `$src` variable, the `htmllib.tcl` package will change the text in the label from the generic image to the actual URL (making it easier for us to decide whether we want to see this image).

```tcl
proc HMset_image {win handle src} {
    bind $handle <ButtonPress-1> "HMimage_request $win $handle $src"
    HMgot_image $handle $src
    return ""
}
```

The Tcl command for creating an image is the `image` command. Tcl can create either bitmap or full-color (photo) images from X11 bitmaps, GIF, or JPEG format data. Several extensions to Tcl add support for other image formats. (See Jan Nijtmans’s site: <http://www.worldaccess.nl/~nijtmans/>).

The Tcl `image` command can accept photo data as binary data in a file, or as BASE64-encoded data in a variable. Since converting binary data to BASE64 and back to binary is a bit silly, we might as well use an intermediate file to hold the binary data.

**Syntax:** `image create type ?name? ?options?`

- `image create` Create an image object of the desired type, and return a handle for referencing this object.
- `type` The type of image that will be created. May be:
  - `bitmap` a two-color graphic.
  - `photo` a multicolor graphic.
- `?name?` The name for this image.
- `?options?` Options that are specific to the type of image being created.

With the `-channel` option, the `http::geturl` command can be instructed to download the URL data into a channel instead of into memory.

Here’s the code for `HMimage_request`. It uses the `pid` command to get the task’s process ID as a cheap and dirty way to make a (probably) unique file name. Once it has a name, it opens an output channel to that file and invokes `http::geturl` to copy the image data from the remote site into that file. Once the image has been created (the data has been read), that file can be deleted with the `file delete` command.

```tcl
proc HMimage_request {win handle url} {
    set tmpFile /tmp/tmp.[pid]
    set outf1 [open $tmpFile w]
    http::geturl $url -channel $outf1
    close $outf1
    set fail [catch (image create photo -file $tmpFile) img]
    file delete $tmpFile
    if {$fail} {
        return ""
    }
    HMgot_image $handle $img
    return ""
}
```
At this point, the htmlview program has most of the functionality of a real browser, with a much smaller footprint than Netscape. Actually writing a full-featured browser in Tcl, however, is not quite so trivial as this makes it look. You may want to look at the Plume browser at <http://plume.browser.org> to see an example of a fully featured Tk-based browser.

A version of htmlview.tcl with support for images and hypertext links is available at <http://www.cflying.com>.

imho: Y2K

1st August, 1999

Dear Mom,

I read the clippings you sent me with dismay. It looks like the press is doing what the press does best — selling fear. Airplanes won't fall out of the sky. Missiles won't fire on their own. Banks won't lose everyone's money. We won't get stuck in 1900, the paint on your house won't turn gray, and your dog won't lose her strong bones and shiny coat. Despite what the press is selling, the wahTOokie bug is nothing more than the modern boogyman: as soon as you look at it, it runs away whimpering. (I've started calling it wahTOOKIE to show how silly it all really is.)

Yes, I know CNN is telling everyone that the banking system will collapse. That's total rubbish! The only reason the banks might have a problem is if people panic and pull their money out. If enough people listen to CNN and their ilk, we could have a problem, yes. Everyone pulling their savings out of the banking system could lead to some serious financial problems. However, that won't be because of Y2K: it'll be because of the press fostering panic. Remember how Hearst got the U.S. into the Spanish-American War just so he could sell more papers? Yellow journalism is alive and well.

I've been looking at this problem for the past several years. People who are relying on ancient systems might have some issues — but most of us have already dealt with (or are starting to deal with) those problems. I fully expect most of the critical systems to continue to function past 01/01/2000. I know the systems I'm responsible for will. I've installed the vendor patches and tested the responses of the computers for all the critical dates (01/01/2000, 02/29/2000, 03/01/2000, 01/01/2001, 02/28/2001) plus, for our people who deal with financials, I've tested quarter-end and year-end dates. The UNIX hosts I've tested have all shown that they will have no problems.

(Yes, there will be a 2/29/2000. I thought we had twisted algorithms, but "every 4 years, except on 00 years, except every 400 years" is just too much fun.)

Of course, some people are still running old OSes or haven't installed the patches their vendors say need to be installed. Those people are, IMHO, foolish. If they're responsible for mission-critical systems, they're not only foolish, but unprofessional in the extreme.

Patching may be a bit of work if you have a lot of systems to deal with, but that's the beauty of UNIX — it can be remotely administered. We have a system that lets us patch all 3,000 of our systems fairly quickly. The mainframe boys have (supposedly) been patching their systems as the vendors put the stuff out. If I were to be worried about anything, it's all those little desktops that are so hard to reach remotely. They're going
to be a big pain. However, very few of them are mission-critical, so it's not as bad as it sounds.

On the plus side, one of the beauties of Y2K is that management is scared of it. Have some old hardware you really need to get rid of? Chances are very good that you'll be saying, "It's not Y2K-compliant, we have to replace it."

I know you're running Windows on your home system. Have you installed the latest patches from Microsoft? I'm glad you finally got rid of that DOS/Win3.1 system. I don't think any amount of patching will fix that old stuff. What did you do with your old 486 system? I hear they make great doorstops. Don't forget to make sure your financial software is up to date. Check your vendor's Web site. (Mine is offering an upgrade for the cost of the CD. Not the latest release, mind you, but one that gets the job done and is compliant. Works for me.)

I guess, if you're really desperate to use that old thing, you could set the clock back to 1972. That year has the same date/day layout as 2000 will. Sure, it's not elegant; consider it a coping strategy. Problem is, some PCs won't let you roll the clock back that far. Of course, those systems are so obsolete they really need to be replaced anyway. I just saw some AMD K6 systems for less than $200. I might upgrade my home system even though I don't need to.

As for stockpiling food and stuff, you should already be doing that. You live in hurricane country! I stockpile stuff for The Big One (it's the price I pay for living in California), and you should be storing enough stuff to get through your local natural disaster. (I'd think a two-week supply would be adequate, if not overkill.) You know, people in the northeast U.S. have to live through two-week power outages when the ice snaps their power lines. I bet they're ready for anything that might come.

Ah, sure, some power generators might go offline briefly. You've dealt with brownouts before, haven't you? Just make sure you have plenty of ice in the freezer and don't open it if you don't need to. If a blackout happens, move some of the block ice into the fridge. Your food will last a couple of days unless you have one of those Florida January hot spells. Besides, I hear ice boxes are retro-chic this year.

How am I preparing for wahTOOkie? I'm keeping my natural-disaster kit up to date, a few weeks' worth of spending money handy (I have traveller's checks and a little cash in my ND kit), and ice in the freezer.

Most important, don't panic. You'll be fine.

Your loving son,

Lee

P.S. Dad was kidding about that bomb shelter, wasn't he?
FreeBSD
Tracking Stable

This article, a practical guide to using and updating FreeBSD on your production systems, should help a newcomer to FreeBSD get productive quickly.

Let's quickly present the advocacy, and be done with it: I think that FreeBSD is the best *nix system and that it is particularly suitable as a server. Developers like it because of the skill level in the development group and because of the permissive licensing terms. FreeBSD is reliable under heavy load, more so than many commercial systems, and impressively secure against attacks across the Net.

And FreeBSD’s reliability is more amazing, considering that the latest OS research is going into it. The Berkeley Fast File System keeps getting faster. There are several development projects under way. See more at <http://www.freebsd.org/projects/>.

FreeBSD is not to be confused with NetBSD, which has been ported to many processor types; or with OpenBSD, which focuses on security; or with Linux, which is . . . Linux.

Choose a Version

Running FreeBSD in a production environment requires a bit of care in deciding which version to run and when to update. Unlike the Linux world, with its many competing distributors (Red Hat, Debian, Caldera, and others), FreeBSD is released from a single organization. Versions have in the past been released quarterly, but that has changed to three per year. The instructions for getting and installing them are at <http://www.freebsd.org/handbook/install.html>.

If you want the most stability, run the 2.2.8 version. It has been frozen for months, with the only changes being careful bug fixes. This is fine unless you have the latest hardware and some new card is not supported, or you really need some new feature.

The 3.0 version had a new framework for driver integration (CAM) and support for many new device types.

The 3.1-Stable version was much improved. The boot loader changed to ELF format, and if you followed the instructions, you did not get stranded at the boot prompt!

The 3.2-Stable version, released in time to be distributed to all Monterey USENIX attendees, promises to be very stable. It comes with a nice default user desktop to save you configuration effort.

Versions are announced on the FreeBSD Web site and on the Announce mailing list, <freebsd-announce@FreeBSD.ORG>.

FreeBSD source is kept in CVS, available on the Net, and it is possible to update your system between versions. There are two CVS “development branches.” The Current branch contains the latest bleeding-edge developments and is recommended to the adventurous user or open-source developer. The Current branch is associated with versions 4.n.

The Stable branch contains development that has been proven by months of use by Current-branch developers and is stable enough for general use. The only changes accepted into the Stable branch are bug fixes, entirely new programs, and new drivers. No major overhauls of proven programs and subsystems are done in this branch. Also, Stable offers binary compatibility with previous versions of the C library and core operating-system functionality.

Most users will want the Stable branch. The Stable branch includes the above-mentioned 2.2.8 and 3.n versions.

Thanks to Bob Gray for his excellent ;login: series on “Source Code UNIX.” Bob explains why you will be wanting to use FreeBSD, how to use it best, and what hardware to run it on. Thanks to reviewers Rik Farrow, Bob Gray, and Alfred Perlstein. Thanks to all the contributors in the FreeBSD world. – RL
FreeBSD supplies patches to fix problems identified by CERT advisories. FreeBSD generally is designed with security as a primary goal, so not many such problems arise.

Version Updating
As ever in software development, new features often cause bugs in existing features, and bug fixes often cause or expose other bugs. You want to use the new features, but you are afraid of sorting out the new version's bugs.

A bug is only a problem if it bothers you. There is a Web-based bug-tracking system at <http://www.freshdesk.org/support.html#gnats>, and you will find that most reported bugs simply don't affect you. Maybe they are related to a certain brand of disk controller card or maybe to a feature that you don't use.

Suppose that you have encountered a bug, and it really is a bug, not a misunderstanding, and maybe you have discussed it on the mailing list <freebsd-questions@FreeBSD.ORG>. (See <http://www.freshdesk.org/support.html#mailing-list>.) You have found that there is now a fix in the CVS Stable branch, perhaps by monitoring the Stable mailing list <freebsd-stable@FreeBSD.ORG>. You decide to update, as described at <http://www.freshdesk.org/handbook/stable.html>.

It is surprisingly simple to update your sources across the network using one CVSUp command. CVSUp is a software package for distributing and updating collections of files across a network. It displays its progress in an X window. CVSUp's streaming communication protocol and multithreaded architecture make it most likely the fastest mirroring tool in existence today. When the CVSUp is finished, the entire base operating system, including kernel, stock user-mode programs, and libraries can be rebuilt to a known configuration by typing a single make command.

The update will fix many other bugs that you might not care about, and possibly (unlikely) introduce a bug that you do care about. How anal do you need to be about updating your production server? You could run a test system, but it is hard to test with a real workload. At some stage, you have to get brave and put the updated system online. The FreeBSD team is devoted to helping fix any problems via the mailing lists. Usually a solution will be provided in under 24 hours, most often in two to three hours! The FreeBSD developers are worldwide, so at any hour of day or night the chances are good that someone knowledgeable is ready to help you.

The above-mentioned mailing lists are really useful (though the volume is bothersome, even with a threading MUA). Commercial system vendors don't give you as much visibility into the release decision-making, especially if you are a smaller customer. With FreeBSD, it is discussed openly on the Stable mailing list, and bugs are tracked on a Web page, so you can understand all the pros and cons of changes.

CVS versus Patching
Vendors like Sun often supply patches for specific bugs. Apply patch 1, then apply patch 2, and then bug 1 reappears. You end up applying a “jumbo” patch that includes fixes for both bugs. Sun provides extensive information on the patches, so you can plan your patching activities. Not even Sun recommends installing every fix. FreeBSD doesn't have any other practical option. To patch FreeBSD you would have to monitor the source changes going into CVS and pick just the ones you want. This might be practical for developers, but most of us will update to the latest CVS and take the small chance of encountering an unrelated bug. Most FreeBSD users prefer the CVS approach to patches. But, as I mentioned above, you are taking a finite risk.

The exception to the above is that FreeBSD supplies patches to fix problems identified by CERT advisories. FreeBSD generally is designed with security as a primary goal, so not many such problems arise. See <http://www.freshdesk.org/security/>.
If you have your own private source changes, including patches, then you don't want them clobbered by a CVSup update. It is a bit tricky, but CVSup makes it easier than it could have been. The CVSup FAQ has help for this at <http://www.polstra.com/projects/freeware/CVSup/faq.html>.

Contribute!
Would you like to begin contributing to the open-source community? Find a significant bug (difficult!), determine that it really is a bug, and develop a simple test that demonstrates the bug. Mention it on the mailing list, and if nobody has a quick fix for it, then submit a problem report to the bugs Web page, <http://www.freebsd.org/send-pr.html>.

Testing is not as glamorous as new development, but it is challenging and it makes a very useful contribution to the FreeBSD world. Some people have a nightly update and build, so they are constantly testing the latest source.

Another way to ease into a contributing role is to help with documentation. See <http://www.freebsd.org/tutorials/docproj-primer/>.

Commercial Services (Plug)
Install from CD-ROMs! CD-ROMs from Walnut Creek are the fastest way to get installed initially. Thanks are due to Walnut Creek, <http://www.cdrom.com/titles/freebsd/> for its generous ongoing support of FreeBSD development. After the CD-ROM installation, you can add prebuilt packages from the CD-ROMs or make ports from sources on the CD-ROMs. Disclaimer: I am just a satisfied customer of Walnut Creek.

If you would be more comfortable with commercial support, there are many consulting services listed at <http://www.freebsd.org/commercial/>. Or, if you want to deal with a larger company, I recommend BSD/OS from Berkeley Software Design, Inc., at <http://www.bsdos.com/>.

Mini-Musings
I am not a communist. I believe that a person should be able to enjoy the proceeds of his/her labor. But I am not sure how this can happen in the software world. FreeBSD and the other open-source projects unfortunately compete with some well-respected commercial companies that deserve to stay in business and stay profitable (I am thinking of Sun, BSDI, and SCO, among others). Is it fair that some of the FreeBSD development is subsidized by government research grants and university funding, and ultimately the taxpayer? This means that the commercial companies are forced to support their competition to some degree. Fair or not, the software world is not going to change direction because of my musings. The affected commercial companies may need to shift the focus of their businesses toward integration, documentation, and support.
Back in the beginning of February, I was writing my April Fool article for \[login\]: I pretended that Bill Gates was announcing Office for Linux. Well, on March 11, ZDnet ran an article about rumors surrounding Linux Office (\[http://www.zdnet.com/zdnn/stories/news/0,4585,2224863,00.html\]). My cracked crystal ball was working overtime, turning up musings stranger than reality.

I found out about the rumor from some email that pointed me to an article at Linux Today. Matt Michie, a student at New Mexico State University, muses about the future of Linux, BSD, and Microsoft (\[http://linustoday.com/stories/6960.html\]). I was very impressed, even to the point of turning over this column to him. Matt did not appear interested in writing for a medium that travels at less than the speed of light, so I am safe for now.

In his article, Matt mentioned that a Win32 layer that will run on top of UNIX already exists. According to Matt (and other sources), this library was created for the ports of Internet Explorer to Solaris and HP-UX, but is now in the hands of Microsoft. While Microsoft has officially denied having a Linux Office project underway, they also denied having a Java version of Office in the works — until they decided to abandon the project, which did get announced.

Matt’s prognostications included the notion of Windows over UNIX — specifically on top of FreeBSD. In Matt’s vision, Microsoft would work with a version of FreeBSD, adding the Win32 library so that it would run Windows 9x/NT applications without modifications (well, theoretically at least). They would choose FreeBSD over Linux because the licensing is less restrictive — they would not have to post modifications to the open-source community. Then Microsoft could modify FreeBSD to its own satisfaction, making it different enough that you could not simply put the Win32 library on top of the original FreeBSD and expect it to work.

Diabolical!? Would they do this? I personally doubt it, but it is a reasonable potential counter-strategy for handling the open-source movement. And I would bet that they are actually working on it, just in case. Microsoft will appear angelic as it releases an open-source version of BSD UNIX, gets good press, and perhaps pulls the wool over the eyes of the Department of Justice in the process. Once everyone is impressed, Microsoft goes about mucking up their “open-source” UNIX, making it just as proprietary as anything else they have ever done.

Free *NIX
My pleas for some insight into why there are three different versions of BSD (and one commercial one, BSDI) around have not fallen on deaf ears. A reader of \[login\]: who preferred to remain anonymous explained it to me. You can also read an insider’s (Jordan K. Hubbard’s) version of some of the history at \[http://www.freebsd.org/handbook/history.html\].

So why are there several versions of BSD? In a word, personalities. Everyone just didn’t get along, so in the spirit of true anarchy, they fissioned into different groups. In a democracy, the majority would force the minority to accept its view, and in a dictatorship, the leader’s view (or the boss’s) would prevail. Anarchy does have something going for it.

Bill Jolitz had started the 386BSD project, an attempt to produce an unencumbered version of BSD UNIX. In those days, you could get the source to BSD only if you had a
source license from UNIX Software Labs, part of AT&T. While most of UNIX had been completely rewritten, there were still some portions that had not. Jolitz set about writing one of the key pieces, memory management for an Intel 80386.

Memory management is critical to any part of any multitasking operating system, as well as being a bear to write. Jolitz (who later joined BSDI), had published the source of his memory management modules in issues of *Dr. Dobbs* back in the early ’90s. Coincidentally, this happened about the same time that Linus Torvalds was writing memory management for what would become Linux. Jolitz managed to produce two versions, .0 and .1.

Neither version worked very well, but there were many patchkits floating around that moved the .1 release toward a truly functioning UNIX system. Jolitz had never been easy to work with and withdrew his sanction for the patchkits in 1993, leading to the founding of the FreeBSD project by Nate Williams, Rod Grimes, and Jordan Hubbard. Hubbard negotiated with Walnut Creek CDROM for both CD publication and a fast Internet connection, and in December of 1993, FreeBSD 1.0 was released.

FreeBSD 1.0 was based upon 4.3BSD-Lite (“Net2”), as well as software from 386BSD and the Free Software Foundation. But Net/2 was considered encumbered by Novell, which had bought the license for UNIX from AT&T by this time. Keep in mind that this was after AT&T failed in its lawsuit to prevent any of this from happening — arguing, in part, that anyone who had ever seen UNIX source code was “contaminated” and could not participate in writing another operating system. If you think this is weird, Microsoft has vaguely similar arrangements in its source-code license. Also, BSDI, which was preparing its version of BSD around this time, was forced to change the last four digits of its phone number. (8649 spells UNIX, as well as TOIZ, on your touch-tone dial.) Note that USENIX was not forced to change its offices’ phone numbers ending in 8649.

An “unencumbered” 4.4BSD-LITE version became the basis for FreeBSD 2.0 in November of 1994. Today, FreeBSD is the most popular version of *BSD in use on Intel platforms, although it pales in comparison with the number of Linux users.

So far, the personalities, with the exception of Bill Jolitz, appeared to get along pretty well. But there had already been a pair of schisms. Theo de Raadt had his own ideas about the direction of FreeBSD and allegedly conflicts were developing with other members of the team. But Theo had an edge by being a Canadian citizen. After moving to Canada, Theo founded OpenBSD, which specializes in security. Canadian laws regarding encryption are much looser than US laws, permitting OpenBSD to include strong encryption in its product, as well as to focus on security fixes in general.

The NetBSD group, I am told, is more interested in experimentation than in having a rock-stable version of BSD. So the group of programmers who wanted a version of BSD for experimentation and research headed off in that direction.

The three versions are pretty compatible outside of the kernel. OpenBSD and NetBSD have support for more CPU architectures than FreeBSD, as well as support for obsolete protocols like XNS and OSI. Other differences include different virtual-memory systems, small differences in the IP stacks, device drivers, kernel configuration, console driver, and timekeeping. My anonymous informant quipped that if NetBSD didn’t exist, it would have to be created for those who want to experiment with BSD or run it on a Mac Quadra 600.
Personally, I believe that the diversity is important. Programmers working on each version can compete to have the fastest file systems or IP stack, support for the most architectures, and/or the most stable system. Since these are open-source systems, the improvements may be copied or cloned, leading to improvements in operating systems in general.

For those of you who missed the USENIX conference in Monterey this summer, all attendees were given four different CD cases, containing: OpenBSD 2.5, NetBSD 1.4, FreeBSD 3.2, and Debian GNU/Linux 2.1. USENIX provided grants to each group to support the production of new-release CDs. I have installed a couple of versions and would be interested in comments from someone who has installed all of them (at least once) but who is not already in the Linux or a particular *BSD camp.

Diversity
A couple of alternatives to the Microsoft hegemony have appeared and deserve mention. Many of you may already be aware of them. The first is StarOffice, which I think almost everyone will already know about (<http://www.stardivision.com/>). StarOffice runs on Solaris (both SPARC and x86 versions) and Linux, and it can read MS Office documents. It’s free for personal use (and their definition of such use does fit my home office), or $40 for an individual license. Of course, by the time you read this, Sun may have purchased StarOffice (look at <http://www.sun.com/software/>).

I followed the registration procedures and found I needed to enable cookies from anywhere to get through it. I was a bit miffed to discover that the download, 70MB, was not gzipped, but found out later that gzipping reduced the size by a total of 3MB. On Linux, you will need about 250MB of free disk space to install StarOffice (the original tar file, the untarred version, and 100+ MB for the minimal install).

When I was done, I had something that not only can create PowerPoint-like presentations, but can read email or act as a newsreader as well. I was, er, overwhelmed. This might fit the bill for a lot of people, and it makes a welcome alternative to being forced to use NT and MS Office on your desktop.

There is also VMware (<http://www.vmware.com/>), a virtual 386 architecture machine that can run under Linux or NT. VMware permits you to install NT so that it runs within the virtual machine, so you can be running Linux and have a window on your desktop where you run Windows applications. I have heard various reports about VMware (some good and some about crashes), but have yet to run it myself, as I keep several dual-boot PCs around and have them all connected using a four-port monitor switch. No cutting-and-pasting between windows, but for testing firewalls, for example, a “virtual machine” just doesn’t cut it. Furthermore, the file systems are not yet shared.

Finally, there is PHP3 (<http://www.php.net/>). I ran across this while wandering around SlashDot.org, and wondered what the heck it was. PHP3 is a replacement for CGI scripts that specializes in database queries. Of course, you can do much more with it than access the fourteen different types of databases supported, but this looks like something more than yet-another-language. I suggest that you check it out (if you haven’t already).
The Open Group and IEEE to Develop Joint Revision to POSIX and UNIX Standards

Collaborative Effort to Yield Version 3 of the Single UNIX Specification (press release)

MENLO PARK, Calif. (16 July, 1999) – The Open Group, a leading consortium dedicated to enterprise integration, and The Institute of Electrical and Electronics Engineers (IEEE), Inc., announced today an agreement for joint development of a common revision to the existing Portable Operation System Interface (POSIX) and UNIX specifications.

Under this agreement, The Open Group and IEEE will share joint copyright of the resulting work. The work will replace the existing IEEE Std 1003.1, 1996 version; IEEE Standard for POSIX – Part 1: System Application: Program Interface (API) [C Language]; and IEEE Std 1003.2, 1992 version, IEEE Standard for POSIX – Part 2: Shell and Utilities; and The Open Group Base specifications for the Single UNIX Specification. It is expected that the joint work will also be put forward for adoption by the International Electrotechnical Commission (IEC) and the International Organization for Standardization (ISO).

This unique collaboration, informally known as the Austin Common Revision Standards Group, combines the formal standards process with the industry specifications for the UNIX system. The resulting document set will replace the existing POSIX.1 and POSIX.2 and become the core of Version 3 of the Single UNIX Specification.

“This agreement offers significant benefits to the industry and to end users. The IEEE POSIX specifications and the UNIX specification are significant foundations for today’s IT systems. By combining the two, the industry is assured of this solid foundation continuing, preserving the high value of investments associated with software systems,” stated Judith Gorman, Managing Director of IEEE Standards.

“The aim for the revision project is to write once, adopt everywhere,” said Andrew Josey, Chair of the Austin Group. “Participation in the project presently includes over 120 individuals from over 50 companies, including representatives from the commercial system vendors, the Open Source community, government and academia.”

The joint revision of standards is anticipated to be finalized in the first quarter of 2001. The first draft specifications are now available from The Open Group Web site at <http://www.opengroup.org/austin/login.html>. Detailed information about the project is found at <http://www.opengroup.org/austin/>.

The Austin Common Revision Standards Group has received widespread industry support for its efforts.

“Compaq has always been a leading advocate of industry standards, is playing an active role in the Austin Common Revision Standards Group effort, and is committed to ensuring that Compaq Tru64 UNIX remains open standards-compliant,” said Tim Yeaton, Vice President and General Manager, UNIX Software Division, Compaq Computer Corporation. “Participation in this initiative underscores our commitment to open standards and helping our customers maintain their investment in existing applications. This effort is going to substantially increase the pace at which
corporate customers adopt UNIX as the IT platform for enterprise applications," Yeaton added.

"Development of these key industry standards continues to provide the necessary freedom of choice in selection of systems from competing suppliers both today and, more importantly, tomorrow," said Denis Brown, Vice President, General Manager, Litton PRC and Chair of the Governing Board of The Open Group.

"The Linux Standard Base is pleased to be contributing to the Austin Group. POSIX is an important factor behind the success of Linux, and the Austin Group update is needed to underpin the future of Linux and other POSIX operating systems," said Dan Quinlan, Linux Standard Base.

"Canada has been and continues to be a strong supporter of this work through our participation via the Standards Council of Canada in ISO/IEC JTC1/SC22/WG15. We look forward to the publication of this important work in the very near future," said Doug Langlotz, the Standards Council of Canada.

"Softway Systems has long supported the ongoing work of the IEEE POSIX standards, and their adoption by The Open Group into the Single UNIX Specification," said Jason Zions, Chief Scientist, Softway Systems and Chair of the IEEE POSIX Working Group for System Services. "These documents form the basis for the INTERIX environment on Windows NT and Windows 2000. We look forward to continuing our participation to ensure a common programming environment exists between Windows NT and all Linux and UNIX systems.”

"Lack of true source-level portability is one of the highest hidden costs in software development, and this agreement is a substantial step towards reducing that cost. Producing a single, coherent, reference work for application developers that will be implemented not just on traditional UNIX platforms but on numerous others, including Linux and Windows NT, is something our 8,000 members have long demanded," said Nick Stoughton, Standards Representative at USENIX Association.

The Open Group has been the custodian of the specification for the UNIX system and the trademark since 1993. The effort that led to this transfer was the catalyst for all vendors to make their systems conform to this single definition, a goal that had been elusive in previous harmonization efforts. Today all the major vendors support the Single UNIX Specification and have registered product. For information on registered products see <http://www.opengroup.org/regproducts/>.
by Peter H. Salus

Peter H. Salus is a member of ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He has held no regular job in the past 30 years. He owns neither a dog nor a cat.

The First Computer

Depending on your point of view, any of these men might plausibly be credited. In a previous column, I mentioned Bernard Cohen's biography of Harold Aiken. I've now gone through Scott McCartney's book on Eckert and Mauchly. Cohen writes: "There is general agreement...that Mark I heralded the dawn of the computer age" (p. xii). McCartney says: "John von Neumann didn't invent the computer. The distinction rightly belongs to two men at the University of Pennsylvania, Prespar Eckert and John Mauchly" (p. 5).

George Stibitz's Bell Labs Complex Number Calculator becomes operational on 8 January 1940. It is a fully operational calculating machine with automatic control of its operations.

In 1942, with its arithmetic unit of 300 tubes complete, the ABC is abandoned by Atanasoff when he goes into the Navy.

In January 1943, the Mark I is operational at IBM's Endicott, NY, labs; it will be moved to Harvard in May 1944.

In spring 1945, the ENIAC is "working well."

At this point, let me point out that just what a computer is, is less than obvious. The usual criteria are that it be digital, be programmed, be able to perform basic arithmetic functions, and employ a stored program. Many folks reject the creations of Zuse and Aiken because they were electromechanical, not electronic. Eckert and Mauchly's ENIAC was electronic, but it didn't have a stored program.

And so we arrive at my conclusion: The winner is Maurice Wilkes; the machine is the EDSAC, designed and built at Cambridge (the one in the UK, not the one in MA). EDSAC ran its first calculation on 6 May 1949.

McCartney's fascinating book on Eckert and Mauchly misses this entirely. It seems to flow more from Al Gore's statement (in 1996) that ENIAC was "the first computer in the world" (an indication of the future claim that the vice president had "invented" the Internet) and Bill Clinton's allusion to it in his second inaugural address (January 1997). Read it, but don't believe everything.

By the way, I know of no book on George Stibitz nor on Konrad Zuse, though at least Zuse's autobiography was brought out in English by Springer (1993).

ARPA's Investments
Janet Abbate's history grows out of her 1994 Pennsylvania dissertation. While it is a relatively good read, it also suffers from a mild case of dissertationitis, and under 30 of the references in her 18-page bibliography are from the last five years. Most of what Abbate has to say is correct, except (in my opinion) she goes awry where the OSI/TCP war is concerned. Her academic bias shows on p. 221, where she refers to Hafner and Lyon (1996) as "journalistic" and John Quarterman's (1990) and my (1995) books as not scholarly. My guess is that this means: not published by a university press.
Coffee Beans? Coffee Grounds?

About two years ago I felt inundated by Java books. The same sort of thing has happened this summer. I received nearly two dozen (some, admittedly, second editions) books on Java, the Java OS, Jini, etc. Too many. But here are my thoughts on three of them.

The Java Virtual Machine is an abstract computing machine that enables Java to host applications on other platforms without rewriting or recompiling code. This has been Java’s claim for several years. Engel’s first-rate book explains how the VM works and how to program it. I especially liked the section on implementing languages (Scheme, Prolog, Eiffel, etc.) and the one on compiling languages.

This led me to Sheng Liang’s volume on the Native Interface, another excellent exposition on how to integrate code in C or C++ into Java code and how to pass data types. Did Andy Koenig invent the “traps and pitfalls” genre? If he did, I tip my hat to him. Liang’s 15-page chapter is very fine.

Finally, along comes the Jini spec. Bill Joy put it best: Jini “is designed to bring reliability and simplicity to the construction of networked devices and services.” The Jini architecture is relatively simple. The specification is not a simple read, but it is a rewarding one.

Telecomms

I enjoyed most of Thurwachter a great deal. It’s telecommunications without calculus, and it’s been a long time since I thought seriously about calculus. The book is clearly intended as a college textbook, but don’t let that prejudice you. In our world, if you work with computers, you need to know something about data and telecoms. There’s a respectable chapter on ISDN, but it informs me that there are “two main types of switching: Circuit switching . . . [and] . . . Packet switching” (p. 539). Message switching anyone? DSL and ATM are also MIA.

The chapter on optics is quite good, but why was WDM relegated to “Multiplexing” rather than “Optical Media”?

I have several times bemoaned the lack of a good book on optical networks. Never again. Stern and Bala have written a very fine volume on a variety of topics involving optical fiber and transmission. There’s a lot of math in this one, but if you give the authors your attention and time, it’s well worth your trouble.

Finally, there’s a second edition of Carne’s Telecommunications Primer. Carne covers a lot of ground and does it well. There are nearly 70 pages of acronyms and glossary and a lot of well-presented other information. One area only, OSI/TCP, causes me unease. Carne states, “TCP/IP applications are limited in scope and flexibility; they cannot invoke many of the services that the OSI model envisions” (p. 640). Yep. TCP was designed by folks dealing with real stuff over real networks. They didn’t “envision,” they made it work.

Apology

I have received several notes concerning my criticism of Mobility, in general because I said little about the contents but a lot about the presentation (dimensions, typography, etc.).

While the book may be ungainly, the contents are first rate, and the editors have done a fine job in pulling the work together. There are a number of papers that are “must reads”: Barak and Wheeler, Cabrera, Douglas and Ousterhout, Cheshire and Baker, the late Mark Weiser, Johansen, Van Rennesse and Schneider, in addition to the Perkins. Even if (e.g.) IEEE has issued photo-offset 8.5”x11” volumes, there’s no good reason for ACM/A-W to do so.
2000 USENIX Nominating Committee

by Evi Nemeth
Nominating Committee Chair

<evi@cs.colorado.edu>

The biennial elections of the Association's Board of Directors will be held in the spring of 2000. The current Board has asked me to chair a committee to nominate candidates for the Board. At the time of this writing, the Nominating Committee is:

Evi Nemeth, University of Colorado, Chair
Dennis Ritchie, Lucent Technologies – Bell Labs
Others will be appointed.

Board members are elected for two years and will take office in June 2000. There are eight Board positions:

President
Vice President
Treasurer
Secretary
Board Member at Large (four positions)

All positions are typically filled from a combination of continuing Board members along with new folks.

We are soliciting suggestions for nominations. The Association needs people who are enthusiastic, energetic, and responsible, and who are willing and able to donate considerable amounts of time.

Candidates should have some political skills as well as a technical interest in USENIX. Vision, insight, and the ability to work and play well with others are also on the wish-list. Having a sense of humor is always an attractive bonus. Folks who are overcommitted tend to be no-ops on the board and that hurts the Association a lot; it needs an active and capable board.

Please send suggestions for nominees by late October to: <nominate@usenix.org>

We also invite feedback and comments (which will be kept strictly confidential) on the current Board members.

If you think you might be interested in running for the Board but aren't sure quite what it entails, mail me <evi@cs.colorado.edu> and I will send you a crib sheet for potential candidates that includes a bit of the job description.

Thank you; your input and suggestions are important.

LISA '99

by David Parter
Program Chair, LISA '99

<dparter@cs.wisc.edu>

Since you are reading :login:, you are probably well aware of the LISA conferences — but you might enjoy this reminder anyway. I am very excited about the program for LISA ’99 (November 7-12 in Seattle). As in past years, LISA ’99 includes a three-day tutorial program and three days of technical sessions with multiple tracks: refereed papers, invited talks, the "practicum" track, and "guru in" sessions. The technical program features a wide variety of interesting and timely papers and talks on all aspects of modern systems administration.

I am particularly excited about the
30 Years Since 1969

by Peter H. Salus
USENIX Historian
<peter@pedant.com>

No, I wasn’t careless. I really want to discuss 30 years ago, not 30 years ago in UNIX. Because I see this summer and autumn as the fruit of a number of things that went on in 1969.

In July 1969, Neil Armstrong set foot on the moon, the culmination of John F. Kennedy’s 1961 vision of putting a man on our satellite within the decade.

In summer 1969, Ken Thompson and Dennis Ritchie created what we now think of as UNIX.

And on September 2, 1969, IMP #1 (Interface Message Processor #1) was plugged in in Len Kleinrock’s lab at UCLA.

It’s pointless to try to enumerate the scientific developments that have grown out of NASA’s programs, for things like the robot arm have proven to have influenced medicine, and the materials research has ended up in everyone’s kitchen.

But think of what the folks at Bell Labs effected where the development of computing is concerned. And imagine what other scientific and engineering achievements wouldn’t exist without the impetus of this operating system.

Finally, here we are, in what Alan Greenspan has referred to as an “information society.” But without the UNIX operating system and the ARPANET/Internet matrix, we would be shipping wads of paper around and we wouldn’t be able to process what we receive.

Thirty years ago I was an associate professor at the University of Toronto, where a gracious administration had “given” me a PDP-8 for research purposes. A 12-bit machine designed and created by Bell and de Castro. 4K of memory. Paper tape. It was only in 1975 that I got a DECwriterII and a 110-baud acoustic modem that connected my office to the IBM 360. I’m typing this onto an Ultra 5 in Boston and will soon send it off via my ISDN connection to the USENIX office.

The world has changed.

Our homes have changed.

Bardeen and Brattain invented the transistor on December 16, 1947. It took nearly seven years to work its way into technology: In October 1954, you could buy a Regency TR1 radio with four transistors manufactured by Texas Instruments. Thomas Watson, Jr., bought over 100, telling his executives that “if that little outfit down in Texas can make these radios work for that kind of money, they can make transistors that will make our computers work, too.”

Four years later, IBM produced its first transistorized computer, the 7090. It was five times faster than the tube-powered 709.

All of this is merely to point out how much has changed and how rapidly.

There are many anniversaries this year, but in one of the most remarkable coincidences, Neil Armstrong, Ken Thompson, Dennis Ritchie, Vint Cerf, Bob Kahn, Len Kleinrock, Doug Engelbart, and their many associates wove the strands of time in that summer 30 years ago that yielded us today’s fabric.

Student Research Grants

by Gale Berkowitz
USENIX Deputy Executive Director
gale@usenix.org

Here are two profiles of USENIX Student Research Grant projects.

Intra-Address Space Protection Using Segmentation/Paging Protection by Ganesh Venkitachalam and Prashant Pradhan
Faculty Advisor: Tzi-cker Chiueh, State University of New York at Stony Brook

Introduction

Two major software applications trends call for operating systems support for establishing protection boundaries among program modules that execute in the same address space. First, the notion of dynamic extensibility has prevailed in almost every major software systems area, ranging from extensible database systems[4] to which third-party data blades can be added to perform type-specific data processing, extensible operating systems[1, 2, 3] that support application-
specific resource management policies, to programmable active network devices[5, 6] that allow protocol code running on network devices to be tailored to individual applications. A key feature of extensible software systems is its support of live addition and removal of software modules into a running program. Therefore, an effective and efficient mechanism to protect the core of the running program from dynamically inserted modules is crucial to the long-term viability of extensible systems. Second, component-based software development (CBSD)[7] is emerging as the dominant software development methodology, because it significantly improves software productivity by encouraging modularity and reusability. As software components produced by multiple vendors are used to construct complete applications, a proper level of protection among software components is essential to address the most challenging problem of CBSD: interference among separately developed components and the resulting system instability. With appropriate intercomponent protection, it is easier to isolate the buggy component and pinpoint the cause of software malfunctioning.

The Palladium project, sponsored by a USENIX Student Research grant, aims to develop an intra-address space protection mechanism by exploiting the virtual memory protection hardware at the paging and segmentation levels built into the Intel x86 processor architecture since 80836. Although a number of approaches have been proposed to provide intra-address space protection, such as software fault isolation[8], type-safe languages[9], interpretive languages[9], and proof-carrying code[10], there is no clear winner that completely solves all the design issues: safety from corrupting extension modules, run-time performance overhead, and programming simplicity. The commonality among all the above approaches is the use of software-only techniques to create protection domains within an address space, based on the assumption that hardware-based protection mechanisms are only applicable to inter-address space protection. In contrast, Palladium's intra-address space protection mechanism is efficient in terms of run-time overhead, guarantees the same level of safety as using separate address spaces, and does not increase the complexity of the deployment and development of software extensions. Although the proposed mechanism is geared towards the Intel x86 architecture, the fact that the architecture in question dominates more than 90% of the world's desktop computer market ensures that it will have wide applicability and thus significant impact.

Approach

The current Palladium prototype is implemented under Linux. A user-level Linux process can dynamically load an extension module into its address space using dlopen, dlsym, and dlclose functions. There is a potential protection issue between the main application, such as a database management system, and the dynamically loaded extension modules, such as type-specific access methods, because they reside in the same address. Note that we focus mainly on the protection between the main application and its extensions, rather than on the protection between extensions.

In x86 architecture, a piece of code or data resides in a specific segment at a specific protection level. There are four protection levels, 0 being the most privileged and 3 being least privileged. x86 architecture also provides a two-level page-granularity protection, with the following mapping rule: segmentation protection levels 0, 1, and 2 are mapped to page protection level 0, and segmentation protection level 3 to page protection level 1. A code can access data only at the same or less privileged levels. Similarly, a code can transfer control only to another code that is at the same or less privileged levels. In order to transfer control from a less privileged to a more privileged level it must go through an explicit interrupt, which allows the kernel to step in and performs necessary checks.

Existing operating systems on the x86 architecture, such as Linux and Windows NT, typically run the kernel at protection level 0 and user-level applications at protection level 3. Palladium, on the other hand, puts the main application in a segment at protection level 2 and the extension modules in another segment at protection level 3. These two segments cover exactly the same address space range, 0 to 3 Gb, but at different paging and segmentation protection levels. As a result, when an extension module attempts to access pages in the main application, the access could go through the segmentation check (because it uses its own segment descriptor), but it would be stopped by the page-level protection check, because the extension's page protection level is less privileged than that of the target page, which is owned by the main application. On the other hand, the kernel segment spans from 0 to 4 Gb. Therefore, although the pages in the kernel address space (3–4Gb) and the user address space (0–3Gb) are all at page protection level 0, the main user-level application cannot access the kernel address space because of segment-level length check. In the end, a user-level extension can only access its own code, data, and stack, as well as shared libraries and data regions exposed by the main application. To summarize: The segmentation check ensures that the kernel is protected from the applications, and the paging check protects the applications from their extensions — exactly the protection properties we are looking for!

To use Palladium's user-level extension mechanism, the main application has to use a safe version of the dynamic loading package, i.e., seg_dlopen, seg_dlsym, and seg_dlclose, to load, access, and close shared libraries. However, seg_dlsym should be used only for acquiring function pointers. To obtain pointers to data structures inside an
extension segment, dlsym should be used instead. In addition, the main application should call the init_pl function in the beginning of the program to promote itself to segmentation protection level 2 and should mark all its pages as page protection level 0. To expose shared pages to extensions, the application can use the set_range system call to mark those pages as page protection level 1. To expose an application service that extensions could use, the application uses the set_call_gate system call to set up a call gate with a pointer to the corresponding application service function.

Programming user-level extensions is exactly identical to developing a user-level library routine, except that xmalloc instead of malloc should be used to ensure that it’s the extension segment’s heap that is being allocated. Palladium’s extensions are compiled with GCC, just like conventional shared libraries. Calling an extension function from an application and returning from a called extension back to the calling application follow exactly the standard C syntax, although applications and extensions reside at different privilege levels.

Current Status

As part of his Master’s thesis, Ganes Venkitachalam has built a working user-level extension system based on the Palladium architecture as described above. To demonstrate the utility of this system, he also built a LibCGI system that allows CGI scripts to be invoked as local procedure calls in a safe fashion, rather than through inter-process communications. Prashant Pradhan has successfully constructed a kernel extension mechanism based on the same segmentation hardware but slightly different software architecture. He has applied this kernel-extension mechanism to his Ph.D. dissertation project, a cluster-based high-performance programmable network router. Prashant is currently working on integrating the user-level and kernel-level extension mechanisms into a fully operational Palladium prototype, which will be part of the Stony Brook Linux Distribution (SLD).

A paper that describes the initial design and implementation of Palladium appeared in the 1999 HotOS Workshop. Another paper that describes the LibCGI system in more detail, including how it exploits Palladium’s user-level extension system, is scheduled to appear in the 1999 IEEE Workshop on Internet Applications. Both papers are available on the project’s Web page: [http://www.ecl.cs.sunysb.edu/extension.html](http://www.ecl.cs.sunysb.edu/extension.html).

Despite the fact that a rich set of protection hardware features have been built into Intel x86 architecture since the days of the 80386, to the best of our knowledge no operating systems or software applications that effectively exploited these hardware mechanisms have been reported in the literature. Palladium is, if not the first, one of the first successful attempts, and it completely solves the protection problem in the emerging extensible and componentized software systems.

Reference


USEWebNET and PaperFinder
By Thanos Papathanasiou
Faculty Advisor: Evangelos Markatos
Computer Architecture and VLSI
Group at the Institute of Computer
Science, Foundation for Research and
Technology Hellas (FORTH) and the
University of Crete.

Overview
USEWebNET and PaperFinder are two valuable tools that aim to help scientists
and all interested users in finding useful information available on-line on the
World Wide Web. They address the problem of “information overloading” by
keeping track of the newest information that gets published on the Web about
registered subjects and provide the opportunity for each user to create a per-
sonal profile with his own registered interests. USEWebNET may be described
as a general purpose meta-search engine that operates on top of several popular
search engines and filters the results in order to avoid presenting the users with
already known information. PaperFinder is an extension of USEWebNET that
retrieves scientific publications from

well-known digital libraries. It also
supports a Resource Discovery Mode of
operation, which broadens a keyword-
based query in order to reveal more use-
ful information, then filters the results
through an innovative similarity metric
of documents.

Information retrieval on the Web is like searching for a needle in a haystack: one
needs the right tools (e.g., a metal detector) to separate the needle from the hay.
According to the traditional model of searching for information on the Web,
when a user wants to find information about a specific topic(s) he sends a query
to a search engine, which replies with several URLs. Every time the user wants
to find new information about the same topic, the search engine returns (roughly)
the same URLs, flooding the user with unnecessary information. Finding the
new and interesting URLs within a slow-moving flood of previously visited URLs
is a boring and time-consuming task.

Assume, for example, that a user is interested in learning new developments in
the field of Web caching. A search for this information using a popular search
engine like HOTBOT will return more than 83,000 URLs. Of those URLs, only a
small percentage (recently, 63 of the
83,000) were created/modified within the
last month. Under this search model,
users who are interested in getting only
new information must process 83,000 URLs in order to extract the few recent ones. One could argue that recent information can be retrieved effectively by requesting only documents that have been published/changed after a specific date (an option with several search engines). Unfortunately, this approach may still result in a document flood.

Since the Web is growing at an alarming rate, the robots associated with search engines visit (and index) various sites rather infrequently. Popular Web robots such as AltaVista visit their archived sites every 2 to 3 months. As the amount of information available via the Web grows larger, this interval is bound to increase to perhaps once every 5–6 months. Thus, if a user wants to find previously unseen information on a specific topic (s)he would have to search for documents that are less than six months old. To return to our example: Such a search on HOTBOT for documents on Web caching returned more than 30,000 documents.

The core of the search problem is that all these engines represent single-shot search mechanisms. A user who repeatedly searches using the same keywords is flooded with almost the same URLs, even if (s)he has visited the URLs as a result of some previous search. This single-shot search contrasts with modern methods of knowledge discovery based on re-searching. Re-searching is an iterative process that filters away useless or already acquired knowledge, focusing on new and unexplored territory. USEwebNET is a layer of software on top of existing search engines, filtering away previously seen information, providing a service that allows users to stay informed of new developments.

USEwebNET consists of a front end that interacts with users and a back end that interacts with search engines. Users register their interests with USEwebNET as a set of keyword-based queries. For example, people interested in caching mechanisms for the Web may register their interest as “Web caching.” In addition, people indicate several search engines they would like to query about their interests. The user interface of USEwebNET is based on the popular and friendly interface of USENET News (Figure 1). Every registered query is used like a newsgroup and new URLs are equivalent to unread articles in a newsgroup.

Periodically (usually every night) the back end of USEwebNET submits each query to the indicated search engines, which reply with a (usually long) sequence of URLs and a short description for each URL. USEwebNET gathers all replies, merges them, deletes duplicates, and constructs a list with URLs that satisfy the query. It then removes from the list all URLs that have been viewed previously by the user.

When the user is interested in seeing recent developments on a given query, (s)he invokes USEwebNET and is presented only with new URLs. The user may decide to view a URL or may mark it as uninteresting and delete it. In either case, USEwebNET will consider the URL as viewed and will not present it to the user again. Viewed documents will be shown again to the user only if USEwebNET detects that an update has been made. To facilitate acquisition of knowledge, USEwebNET allows users to save URLs in folders. Thus, users can store all interesting URLs and visit them at some later time, or use them as references. Over time, folders will eventually become indispensable reference tools for research. In order to help users check the results of their registered queries, USEwebNET maintains a mail notification system, which, when activated by the user, sends an email notification whenever a significant amount of new information has been discovered. Finally, the AT&T Difference Engine [1] has been integrated into USEwebNET’s system in order to help users keep track of modifications to especially interesting Web pages.

**PaperFinder Described**

USEwebNET has been designed as a layer of software that runs on top of existing information providers. Its purpose is to discover and filter information in a specific subject. It may be customized to work with specific databases in order to find information more effectively. This idea has been previously stated by M. Schwartz [2]:

Information extraction is most effective when exploiting the semantics of particular types of files and particular execution environments.
Thus, USEwebNET was extended in order to take advantage of the particular characteristics of scientific publications and the particular services provided by popular digital libraries. These extensions lead to the creation of PaperFinder, a research paper discovery tool, which operates on top of digital libraries and aims to help scientists staying informed about the latest evolutions on their field. Scientists always want to stay informed. In order to do so they subscribe to scientific journals, go to conferences, collaborate with colleagues, etc. To narrow down the information they receive, scientists subscribe only to a small subset of journals and follow only a small number of conferences. Unfortunately, the number of scientific publications increases year after year, making it increasingly harder for a single person to keep track of published papers in a field. Thus, a tool that could deliver to scientists only the interesting research papers in their field would be very useful.

PaperFinder meets these requirements and comes very close to this ideal tool. Currently, most of the scientific publishers provide on-line databases with the titles, authors, abstracts, and sometimes even full text of their publications. PaperFinder can be used as a research paper discovery tool on top of several such databases. To use our earlier example, a scientist may submit to PaperFinder that (s)he is interested in “Web caching.” PaperFinder will continually monitor the research paper databases to find papers that match the query. If such matches are found they are stored in a database. When the user invokes PaperFinder, (s)he will view the new papers found. After the user reads these papers they will not show up again unless the user saves them in a folder. Effectively, the user registers his or her interests with USEwebNET, and the tool continually delivers new research papers found on this field without delivering the same paper twice.

Besides the classic keyword-based way of retrieving information from databases, PaperFinder provides an innovative Resource Discovery Mode of operation. The purpose of this mode is to collect as many papers as possible about a topic and help the user identify those of primary interest.

In the Resource Discovery Mode, the user indicates one or more “seed papers” and expects PaperFinder to discover new articles that are related to them and/or sort the retrieved papers with respect to the seed papers. PaperFinder uses query generalization and filtering to discover papers related to the seed paper(s).

**Query Generalization**

The goal of this step is to find papers that are related to the seed paper. This may be accomplished by forming queries whose results should return the seed papers as well as several other papers. Methods for generalizing an initial query include the following:

(i) Keyword extraction from the titles, abstracts, or/and text of the seed papers. This method complements the initial keyword-based query with more keywords or key phrases, which may lead to the retrieval of a larger amount of useful and interesting information.

(ii) Use of bibliographic references. Every scientific publication contains a “Related Work” section. Conversely, a research paper forms the basis for future developments and is cited by future articles. The citations of a seed paper as well the articles that cite the seed paper provide a large number of interesting articles that should be presented to the interested researcher. A tool that follows this method is CiteSeer[3, 4].

(iii) Use of subject descriptors. A subject descriptor consist of a small number of keywords or/and key phrases that characterize the research area in which a publication belongs. An article may belong in several different research areas. Articles that have at least one common subject descriptor may be related and should be presented to the user. This method is used by the Digital Library maintained by ACM[5].

(iv) Use of "seed authors." Authors who tend to cooperate in publishing scientific papers have similar research interests. Thus, retrieving the articles of authors who are scientifically related to a specified seed author may reveal new interesting papers.

The current implementation of PaperFinder supports a Query Generalization method that is based on a service provided by ACM’s on-line Digital Library, through which papers belonging to the same subject descriptors (maintained and classified by ACM) as the seed paper are downloaded.

**Filtering**

The goal of this step is to filter (sort) the papers found in the previous stage and return the most relevant to the user. Finding which papers are relevant can be tricky and result in a flood of irrelevant publications. To find relevant papers PaperFinder applies a similarity metric to the papers.

The similarity metric currently used by PaperFinder is a sorting algorithm based on author-distance, a notion that originated from the Erdos number[6]. Paul Erdos, who was a brilliant and widely traveled Hungarian mathematician, wrote hundreds of mathematical research papers in many different areas, many in collaboration with others. His Erdos number is 0. His co-authors have Erdos number 1. People other than Erdos who have written a joint paper with someone with Erdos number 1 but not with Erdos have Erdos number 2, and so on.

Based on the Erdos number, we define a similar metric we call author-distance.
Two authors have an author-distance number of one if they have co-authored at least one paper. Their author-distance is two if they are not co-authors but there exists at least one third author who has written a paper with both of them, and so on. The distance between two authors who are not co-authors is found by the transitive closure of the distances among co-authors.

Using author-distances, the scores used for sorting the papers are calculated as follows. The sum of the author-distances among a retrieved paper’s authors and each of the seed paper’s authors is computed and the arithmetic mean is calculated. A small arithmetic mean shows that the respective publication is closely related to the seed paper. The rationale behind this approach is that closely related articles are expected to have authors who have similar interests and some of them probably have co-authored a paper.

Project Status

Two fully stable implementations of USEwebNET and PaperFinder have been developed. Both tools have been installed on a server at the Institute of Computer Science—FORTH and have been operational for the past six months. Using USEwebNET for this time period has shown that people prefer to use USEwebNET instead of directly querying search engines, mainly because of USEwebNET's ability to retain the history of each query, automatically download new results about a registered topic, and provide several useful operations for each retrieved Web page.

Currently, experiments are being conducted to evaluate PaperFinder's Resource Discovery Mode of operation. The first experimental results have proved that PaperFinder's sorting algorithm is able to provide at least as good results as those provided by the digital library of ACM. However, there is still room for improving the filter. For example, at least two other distance metrics may be used to improve our filtering algorithm:

1. Weighted-author distance: The distance between two co-authors is defined as the number of papers they have co-authored over the total number of papers they have authored. The distance between two authors who are not co-authors is found by the transitive closure of the distances among co-authors.

2. Keyword distance: Given two papers and a set of keywords, the distance between the papers is defined as the number of keywords that appears in (the title/abstract/text) of both papers.

In addition, new ways of exploiting the ideas of seed authors (papers) and author-distances should be explored. The first experimental results showed that PaperFinder has the ability to identify the work of distinctive research groups just by using one of their publications as a seed paper in the filtering process, which in turn leads to the idea of identifying clusters of cooperating research groups and exploiting links (publications) among them in order to find articles that couldn't be discovered by other search methods.

Implications

We view USEwebNET and PaperFinder as value-added services on top of existing information providers. They offer several general advantages:

- They capitalize on the well-known and friendly user interface of USENET News.
- They filter information so that users are able to focus on what’s new in their field of interest.
- They help users discover new information.
- They run off-line at night, when communication costs are low. Finally, by running at night they offload busy Web servers and proxies on the Internet.

PaperFinder provides additional services:

- It filters available information, presenting it in order of importance (relevance to the user's topic of interest).
- Its filter is updated periodically and evolves as new information becomes available.


Two papers (a refereed and a short paper) concerning USEwebNET have already been published [7, 8]. We hope to present USEwebNET and PaperFinder at a future USENIX conference. The most suitable conference for this purpose would be USITS'99, where both tools could be presented in a Works-in-Progress report.

Contact Information

Athanasios E. Papathanasiou: ICS-FORTH; tel: +30 81 391437; email <papathan@ics.forth.gr>.

Evangelos P. Markatos: ICS-FORTH; tel: +30 81 391555; email <markatos@ics.forth.gr>.

References


Meet USENIX New Staff

We are delighted to announce two new members of the USENIX team: Monica Ortiz and Moun Chau both joined the staff this summer. Here's what they have to say for themselves:

Before joining the USENIX Association, Monica Ortiz was immersed in the waters of corporate America, doing marketing and project management for CIBER, Inc., Computer Resources Group, and Analog Devices. Her professional experience includes designing and implementing comprehensive marketing plans and projects, advertising, trade show and job fair event planning, and database analysis and reporting. As part of the USENIX team, she will be responsible for the marketing department tasks for USENIX and will be permanently camped out at the Executive Offices in Berkeley.

More important, she meets the new employee height requirement of being under 5’4”.

Production editor Moun joins the USENIX staff with a B.A. in Cognitive Science from the University of California, Berkeley. Her work background includes rising far too early to staff donut shops, roaming campus as a security monitor, and repairing tidbits in the field of electronic publishing. Off work she can be found in arcades playing Area 51.

[Editor’s note: Yes, she too meets the height requirement. We have standards to maintain...]

USA Computing Olympiad Picks Final Four

by Don Piele

Director, USACO
<piele@cs.uwp.edu>

On June 15, fourteen students from around the United States traveled to the University of Wisconsin – Parkside to participate in the seventh USA Computing Olympiad Training Camp. Their goal – to become one of four students to represent the United States in the Central European Olympiad in Informatics in Brno, Czech Republic, and later in the International Olympiad in Informatics in Antalya, Turkey, October 9–16. These final fourteen students were chosen from over 300 participants who had entered the three Internet competitions and the National Championship held in April.

The fourteen finalists were: David Cheng, Junior, Brandywine HS, Wilmington, DE; John Danaher, Junior, Thomas Jefferson HS for Science and Technology, Alexandria, VA; Gary Huang, Sophomore, Appleton West HS, Appleton, WI; Bill Kinnersley, Junior, Lawrence HS, Lawrence, KS; Percy Liang, Junior, Mountain Pointe HS, Phoenix, AZ; Benjamin Mathews, Senior, St. Marks HS, TX; Jon McAlister, Senior, Langham Creek HS, Houston, TX; Ilia Mirkin, Sophomore, Thomas Jefferson HS for Science and Technology, Alexandria, VA; Oaz Nir, Sophomore, Monta Vista HS, Saratoga, CA; Vladimir Novakovski, Freshman, Thomas Jefferson HS for Science and Technology, Alexandria, VA; John O'Rorke, Junior, Centennial HS, Boise, ID; Kaushik Roy, Senior, Montgomery Blair HS, Silver Springs, MD; Daniel Wright, Senior, St. David's College (South Africa), Lafayette, CO;
Daniel Zaharopol, Junior, Vestal Senior HS, Vestal, NY. Also attending the camp as a guest was Mathijs Vogelzang, Senior, The Netherlands.

**All New Training Materials**
Preparations for this year's camp began in April when head coach Rob Kolstad and Hal Burch met in Colorado Springs over Hal's spring break and began studying past IOI competitions. After studying past competition, they identified sixteen problem types that appear in programming contest problems. Among those sixteen, they found that, 80% of the time, only a handful were used. They also discovered that when several different algorithms are combined into one problem, it becomes much more difficult to get the problem right.

Armed with this knowledge, Rob and Hal began creating a personalized training program for the participants, based on which algorithms participants knew and on which ones they were weak. This was determined by a diagnostic exam given at the first session. It was quickly discovered that dynamic programming was a universal weakness among the participants.

During the eight-day training program, topics such as “Crafting Winning Solutions” by Greg Galperin, “Graph Theory” by Brian Dean, and “Debugging & Test Data” by Russ Cox were presented by the other coaches.

The first challenge round was held on the fourth day of the camp. It was the traditional five-hour competition in which each participant worked on finding solutions to four problems: Bessie's Gambit, Alfa Beta Field Forever, Bull Dozing, and Playing Herd. Every problem had been "cowified" in keeping with our Wisconsin dairy farm tradition. This allows the staff a creative outlet unparalleled in other academic competitions!

**Automatic Grading**
The grading was done automatically and at a distance by an ingenious program created by Russ Cox. Russ sent the source code for each program over the Internet to his PC back at Harvard, which contained his grading program. Everything worked as planned, and within a couple of hours all programs were graded and returned. His program revolutionized the way we graded all programming competitions this year and the way we'll go in the future.

**Recreation**
For recreation, we visited the student union rec room on the first night, where we enjoyed unlimited bowling, pool, and ping-pong. Ultimate Frisbee was the game of choice for relaxation. In addition, this year the participants were treated to a private lesson in Disk Golf and played a newly installed eighteen-hole course on the UW–Parkside campus. In the evenings we played Mancala and a business simulation game. Participants wrote programs to play Mancala, which they played against each other, with the moves projected onto a big screen for all to admire.

**Special Events**
The jeopardy-like quiz show complete with multimedia sound and graphics, hosted by Rob Kolstad, was a special treat. This event got even the most serious of computer nerds to laugh—a major accomplishment. Movie night, a picnic in the park, and a picnic at the beach of Lake Michigan helped relax the participants.

The second day of competition followed the same pattern as the first. Once the scores were back from the automated grading program, we combined the results from the two days and ranked them all. As usual, the top three were easy to spot but the fourth was not. After a discussion of the merits of those who were close, a vote was taken and the final four selected. They were, in alphabetical order: David Cheng, Percy Liang, Ben Mathews, and Daniel Wright. The first alternate was Jon McAlister.

The awards banquet followed at a restaurant on the Lake Michigan shore. Everyone received an individual picture, a group picture, and a finalist or team member certificate. The top four each received a personalized trophy. Of course, we could not resist handing out special certificates immortalizing a particular behavior unique to each participant.

The following day we made a visit to the dairy farm owned by farmer Jim. He was a replacement for farmer Paul, whom we had visited for the last four years. Unfortunately, after three generations of milking farmer Paul has decided to give up dairy farming. A wind storm last October had blown down a small barn and a silo, and that, coming on top of falling milk prices, was the last straw.

Then it was off to Six Flags Great America and the chance to ride "Raging Bull," the newest and most challenging roller coaster. Just trying to get a ride on all the roller coasters in the park was a full-day enterprise.

Once again, the curtain came down on another successful USACO training camp. Thanks to the generous support of USENIX, over three hundred students in the United States and an equal number abroad were challenged by the competitions provided by the USACO. All expenses for food, travel, lodging, awards, polo shirts—you name it—were paid for by a grant from USENIX.

The final four team members have been chosen and have their tickets for the trip of a lifetime to the Central European Olympiad in Informatics in Brno, Czech Republic, and the 11th International Olympiad in Informatics in Antalya, Turkey.
MEMBERSHIP INFORMATION

Indicate the category of membership which you prefer and send appropriate annual dues

- Individual: $80
- Full-time Student: $25
  Please enclose a copy of your current student ID.
- Educational: $200
- Corporate: $400
- Supporting – USENIX – $1000 / $2500
  SAGE – $1000

A designated representative for each Educational, Corporate, and Supporting membership receives all USENIX conference and symposia proceedings published during their membership term plus all member services. Supporting members also receive one free full page ad in *login*, more member-rate discounts for technical sessions, a half-price rental of the mailing list, 10% discounts on vendor displays booth rentals, and a link to their URL from the USENIX Web site.

$50 of your annual membership dues is for a one-year subscription to *login*.

SAGE, the System Administrators Guild

SAGE, a Special Technical Group within the USENIX Association, is dedicated to the recognition and advancement of system administration as a profession. To join SAGE, you must be a member of USENIX.

- Individual: $30
- Students: $15

MEMBERSHIP APPLICATION: □ New □ Renewal

Name ___________________________________________________________
City: ___________________ State: ___________________ Zip: __________ Country: _________________________
Phone: _______________ Fax: _______________ Email: ________________________________

MEMBER PROFILE: Please help us serve you better!

By answering the following questions, you provide us with information that will allow us to plan our activities to meet your needs. All information is entirely confidential.

What is your job function?
1. □ System/Network Administrator
2. □ Consultant
3. □ Academic/Researcher
4. □ Developer/Programmer/Architect
5. □ System Engineer
6. □ Technical Manager
7. □ Student
8. □ Webmaster
9. □ Security

What is your role in the purchase decision
1. □ Final
2. □ Specify
3. □ Recommend
4. □ Influence
5. □ No role

□ I do not want USENIX to email me notices of Association activities.
□ I do not want my address made available for commercial mailings

PAYMENT OPTIONS:

Total enclosed $ _______________

□ I enclose a check/money order made payable to USENIX Association

□ Enclosed is our purchase order (Educational, Corporate, and Supporting members only, please).

□ Charge my: □ Visa □ MasterCard □ American Express

Account # _______________________________________________ Expiration Date ________
Name on Card ________________________________________________
Signature __________________________________________________

Outside the USA? Please make your payment in US dollars by check drawn on US Bank, Visa/MasterCard, International postal money order

USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, California 94710 • Phone: 510 528 8649 • FAX: 510 548 5738 • Email: office@usenix.org

revised 9/9/99
Workshop on Applications of Embedded Systems

Sponsored by USENIX, The Advanced Computing Systems Association and the MIT Media Laboratory

March 20-22, 2000

San Francisco, California, USA

Important Dates:
Submissions due: November 15, 1999
Notification to authors: November 29, 1999
Registration materials available: January, 2000
Final camera-ready papers due: February 1, 2000
Event dates: March 20-22, 2000

Program Committee
Dan Geer, SystemExperts Corp & USENIX
Michael Hawley, MIT Media Lab, Things That Think
Other program committee members to be announced.

Workshop Overview

The PC monolith is breaking down; concentrated “core” elements of computing and communication, sensors and actuators will become embeddable in almost everything. The “jellybean” processors that currently pervade nearly every appliance, yet are utterly isolated, will be connectable through a wealth of emergent capillaries sprouting from the internet. Technologies will be produced that are inexpensive, low-power, and radically different from today’s chip-and-pc-board variety, e.g. printable circuits, wind-up electronics, wearable networks powered by walking or breathing, even edible circuitry. Ingredients like these will form the foundation of a vastly extended network of things that are very different from PC’s. Within ten years, a billion people on line will be joined by a trillion things with embedded networks.

The goal of this workshop is to convene a limited number of leading engineers and researchers from a wide cross section of academia, industry, and government to discuss critical challenges in developing and deploying embedded intelligence over a wide range of applications. These are “out of the box” systems in every way, shape, and form. They demand big, bold, maverick thinking. They also demand that we share what we learn by doing, hence the focus on Applications of Embedded Systems.

This 3-day meeting will consist of invited talks, refereed papers, and work-in-progress reports, and a lot of time to mingle informally.

We hope the results of this workshop will help clarify and coordinate the research and development agenda in embedded systems, recognizing that, in engineering and science, getting the problem statement right is much of the battle. Participants will engage in discussions that will encompass a range of areas from low-level materials innovations to novel forms of networking, new kinds of software systems to groundbreaking applications, usability to high-level policy.

Workshop Topics

Submissions are being solicited in the following areas, including but not limited to:

- Applications in unusual domains: toys, appliances, cars, human implants, domestic, rural, outdoor, undersea
- Capillary network architectures (Bluetooth, IrDA, PLC, etc.)
- Software systems to make these systems work
- Case studies and especially those with cost-benefit analyses
- New interface paradigms
- Self-healing and self-assembling systems
- Drastic scaling issues and localization
- Secure communications

We particularly invite those working now in areas such as:

- Telecom such as cell phones, pagers, PDAs
- Domestic technology such as building control, appliances, toys
Medical applications such as implants, sports/fitness instruments
■ Industrial Automation especially where there is hybridization
■ Automotive Applications

Student Stipends
The USENIX student stipend program covers travel and hotel to enable full-time students to attend. Preference is given to students who are speakers. To apply, see: http://www.usenix.org/students/.

What To Submit
The program committee invites submission of an extended abstract, which should describe original work concerning the design, implementation, and real application of embedded systems. We are not looking for tweaks to Linux, or stuffing WinCE palmtops into toys. Rather, we are seeking radical new architectures, exceptionally promising prototypes, and enlightening case studies. The abstract should convince the program committee that a good paper and 20-minute talk will result. Identify what has been accomplished, why it is significant, and compare it with relevant work in the field. Include references, illustrations, and performance data. Be incisive and cogent. If you do not have work to report, tell us why you should participate in this workshop. Preference for this limited attendance workshop will be given to those who have submitted abstracts.

How and Where to Submit
Email the extended abstract (plain ASCII, HTML, or a URL) by November 15, 1999 to:
est00papers@usenix.org

The extended abstract should be 5-7 pages long or about 2500 words, not counting references and figures. You may also submit a full paper at this time. Full papers will go through a brisk editorial review cycle with the program committee, and should be 10-15 pages long. Camera-ready final papers are due on February 1, 2000 for publication in the workshop proceedings.

All submissions will be acknowledged electronically. If you do not receive word within 72 hours of submission, contact the program chairs: est00chairs@usenix.org.

All submissions will be held in strict confidence prior to publication, but they must not be bound by proprietary or non-disclosure arrangements.

Registration Materials
Materials containing all details of the technical program, registration fees and forms, and hotel information will be available in January 2000. If you wish to receive the registration materials, please visit the workshop Web site or contact:
USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1.949.588.8649
Fax: +1.949.588.9706
Email: conference@usenix.org

Rev. 9/7/99
GOT QUESTIONS?  
WE HAVE ANSWERS!

10% Discount to Usenix Members!  
Please mention code 49214 when ordering.

**Program More Effectively**

**Learn how with these books from Addison-Wesley**

**Coming Soon!**

*The pearls are back!*

**Programming Pearls**
Jon Bentley
0-201-65788-0
256 pages

**The Practice of Programming**
Bruce C. Baer and Gary McWilliams
0-201-61586-X
284 pages

**Larry Constantine**
SOFTWare FOR USE
0-201-92478-1
608 pages

**Coming Soon!**

**Build Strong Networks**

**Interconnections**
Second Edition
Michael D. Smith
0-201-63448-1
560 pages

**Insiders Java 2 Platform Security**
Architecture, API Design, and Implementations
Christopher P. Kruegel
0-201-31000-7
249 pages

**Firewalls and Internet Security**
Repelling the Wily Hacker
William Arbaugh
0-201-63357-4
320 pages

**Web Security**
A Step by Step Guide
Eino J. Korpela
0-201-63489-9
448 pages

**DIGITAL CERTIFICATES**
Applied Internet Security
John Flippin
0-201-30980-7
480 pages

**Special 20% Discount**

for USENIX Association members!

Save 20% on all Professional Computer and Engineering books from Addison-Wesley.
Mention Discount Code 71981 to receive your USENIX Association discount.

Call 1-800-824-7799 and save today!

**Addison-Wesley**

Sign up for our mailing lists!

Your resource for complete, expert, and definitive technical information

**Check us out!**

*Offer expires on 12/31/99*
I've been hit hard this week by various pieces of information that have me reeling. What does one believe?

The FBI announced that maybe they did, after all, use a couple of incendiary cylinders at Waco. They weren't used near the ultimate fires in the main building but in a different building altogether. Wish they'd mentioned that earlier.

Top scientists have announced that maybe some other mechanism can cause AIDS besides the HIV-1 virus. Furthermore, it might have something to do with that old polio vaccine thing from the '60s and its simian virus, SV40. This has been "conspiracy theory" fodder for years. Now it turns out that just maybe there is strong basis in fact.

On the same day that the previous two items appeared in newspapers, word came that top security researchers announced (at a USENIX symposium!) the discovery of a variable in the cryptographic API for Windows operating systems (of all flavors) with the name "_NSAKEY." Speculation abounds.

I have been a real non-believer in conspiracy theories. I really think that Oswald acted alone. I have every confidence in NASA's ability to land people on the moon. Fidel Castro is just a guy with an amazing strong set of political skills. The World Bank is controlled by ... bankers.

But, thrice in one day I was hammered with revelations. I think it's extraordinarily difficult to carry around a few bits of "confidence data" on information one receives. "Well," your mind would catalog, "I heard that from Joe and he's 87% reliable, so I'll color that fact as probably true." Oh my. That would use up far more memory and processing than my little brain could ever produce. I fear that upgrades are too expensive, too (those new smart mice notwithstanding).

Little wonder that so many of our fellow world citizens are very skeptical about what they hear and read. I hear from colleagues in Europe who regard the U.S. government's actions in Kosovo as nothing short of a prelude to invasion of the rest of Europe. I'd like to tell them, "Oh, no, our government wouldn't do that." I'm pretty sure. I wish I were 100% sure.

I see people whose actions are mysterious to me. Are they trying to be clever and use some sort of strange political machinations to accomplish private yet nefarious aims? Heck, I don't know. I don't even know how to find out.

I really like trust. It's important to me. I try to be trustworthy.

But these last few days have harrowed me. Whom do you trust?
I don’t have time for learning curves.

They rely on you to be the 1 expert on tough development challenges. There’s no time for learning curves, so you go online for 2 technical references from the experts who wrote the books. Find answers fast simply by clicking on our search engine. Access hundreds of online books, tutorials and even source 3 code samples 4 now. Go to 5 EarthWeb’s ITKnowledge, get immediate answers, and get down to it.

Get your FREE ITKnowledge trial subscription today at itkgo.com.
Use code number 941.
CONTRIBUTIONS SOLICITED

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

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of ;login: are December 1, 1999, and February 2, 2000.