inside:

SAGE NEWS & FEATURES
Election Results!

STANDARDS REPORTS

BOOK REVIEWS

USENIX NEWS

features:

NEW!

The Tclsh Spot
by Clif Flynt

Electronic Snake Oil
by Matt Curtin

A Management Perspective on
Privileged Access to Computer Systems
by Jim Hickstein

and more . . .
# Upcoming Events

## 5th Conference on Object-Oriented Technologies and Systems (COOTS)
- **When:** May 3-7/99
- **Where:** San Diego, CA
- **Chair:** Murthy V. Devarakonda

## SANS99
- **Co-sponsored by:** SAGE
- **When:** May 9-15/99
- **Where:** Baltimore, MD

## USENIX Workshop on Smartcard Technology
- **Co-sponsored by:** CardTech/SecurTech
- **When:** May 10-11/99
- **Where:** Chicago, IL
- **Chair:** Scott Guthery & Peter Honeyman

## USENIX Annual Technical Conference
- **When:** June 6-11/99
- **Where:** Monterey, CA
- **Chair:** Avi Rubin, Program Chair
  Cem Col & John Heidemann, IT Coordinators
  Jordan Hubbard, FreeBSD Track Chair
- **Deadlines:** Final Papers April 27/99

## 3rd USENIX Windows NT Symposium
- **When:** July 12-14/99
- **Where:** Seattle, WA
- **Chair:** Werner Vogels & Stephen Walli
- **Deadlines:** Final Papers June 1/99

## 2nd Large Installation System Administration of Windows NT Conference (LISA-NT)
- **Co-sponsored by:** USENIX and SAGE
- **When:** July 12-14/99
- **Where:** Seattle, WA
- **Chair:** Gerald Carter & Ralph Loura
- **Deadlines:** Final Papers June 1/99

## Eighth USENIX Security Symposium
- **When:** August 23-26, 1999
- **Where:** Washington, D.C.
- **Chair:** Win Treese, Program Chair
  Avi Rubin, IT Coordinator
- **Deadlines:** Final Papers April 27/99, July 12/99

## 2nd Conference on Domain-Specific Languages
- **Sponsored by:** USENIX in cooperation with ACM SIGPLAN and SIGSOFT
- **When:** October 3-6/99
- **Where:** Austin, TX
- **Chair:** Thomas Ball
- **Deadlines:** Notification to Authors June 2/99, Papers August 24/99

## 2nd USENIX Symposium on Internet Technologies and Systems
- **Co-sponsored by the IEEE Computer Society Technical Committee on the Internet**
- **When:** October 11-14/99
- **Where:** Boulder, CO
- **Chair:** Fred Douglass
- **Deadlines:** Notification to Authors May 14/99, Papers August 31/99

## 13th Systems Administration Conference (LISA '99)
- **When:** November 7-12/99
- **Where:** Seattle, WA
- **Chair:** David Parke, Program Chair
  Jennifer Katinsky & Phil Scarr, IT Coordinators
  Bruce Alan Wynn, Practicum Chair
- **Deadlines:** Extended Abstracts May 25/99, Notification to Authors June 30/99, Final Papers September 1/99

## Tcl/Tk: The 7th USENIX Tcl/Tk Conference
- **When:** February 14-18/2000
- **Where:** Austin, TX
- **Chair:** De Clarke & Tom Poindexter
- **Deadlines:** Paper Proposals September 1/99, Notification to Authors October 1/99, Final Papers December 20/99

## Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000)
- **When:** October 23-25/2000
- **Where:** San Diego, CA
- **Chair:** Michael B. Jones & Frans Kaashoek
- **Deadlines:** Paper Submissions April 25/2000, Notification to Authors June 29/2000, Final Papers August 31/2000

For a complete list of future USENIX events, access [http://www.usenix.org/events](http://www.usenix.org/events)
IN THIS ISSUE . . .

LETTERS TO THE EDITOR
3 Corrections

THE USENIX CROSSWORD PUZZLE
3 April Puzzle & February Solution

SAGE NEWS AND FEATURES
4 NT: (The Manual Is) “Not There”! by Phil Cox & Tina Darmohray
6 Broken Paradigm by Hal Miller
7 Election Results by Gale Berkowitz
7 SAGE Certification Subcommittee Briefs
8 Round and Round We Go by Barbara L. Dijkstra
8 The 1998 Outstanding Achievement Award
9 System Profiles with syssum by Bruce W. Mohler
14 Resume Your Resume Writing by David Clark
18 Bandwidth Versus Latency: Helping Developers Understand Network Performance Limitations by Brent Chapman
20 Effective Perl Programming: Analysis Without Paralysis by Joseph N. Hall

FEATURES
25 The Dark Side of Regular Expressions by Andrew Hume
26 The Tclish Spot by Clay Flinn
31 Electronic Snake Oil by Matt Curtin
39 A Management Perspective on Privileged Access to Computer Systems by Jim Hickstein
44 The Webmaster: Password-Protecting Areas of Your Web Site by Dave Taylor
48 Java Performance by Glen McCluskey
51 Using Java: Applets and Animation by Frithvi Rao
56 Biometrics: Untruths and the Truth by Dario Forte
59 Source Code UNIX: Embedding Source Code UNIX in the Product by Bob Gray
65 Musings by Rik Farrow

USENIX NEWS
71 Notice of Annual Meeting
73 The John Lions Award by Dr. Lucy Chubb
73 New Staff at USENIX
74 20 Years Ago in USENIX by Peter H. Salus
75 USENIX Funding Helps Improve the US Patent Process for Software by Cynthia Deno

ANNOUNCEMENTS AND CALLS
76 COOTS ‘99: 5th Conference on Object-Oriented Technologies and Systems
78 1999 USENIX Annual Technical Conference
82 Workshop on Smartcard Technology
84 13th Systems Administration Conference (LISA ’99)
88 motd by Rob Kolstad

STANDARDS REPORTS
68 Open Source – A Standards Success Story? by Nicholas M. Stoughton
69 Is Linux the Future of POSIX? by David Blackwood

BOOK REVIEWS
70 The Bookworm by Peter H. Salus
72 SNA and TCP/IP Enterprise Networking Reviewed by Daniel Lazenby

Cover Photo: The New Sage Executive Committee. From left to right, standing: Hal Miller, Jim Hickstein, Xev Gitter, Timothy Gassaway; sitting: Peg Schafer, Barbara L. Dijkstra, Geoff Halprin
in this issue . . .

This issue seems particularly packed with good advice. Who isn’t concerned about security these days? Phil Cox and Tina Darmorhay tell the tale of Tina’s quest: how to turn off SMB printer and file sharing so that an NT machine can run a Web server with decent security. A long chase and a tough one, but the winner of SAGE’s Outstanding Achievement award never faltered.

When Bruce Mohler needed to automate his collection of system profiles, he turned to Perl and the Web. His sysmap software gathers an impressive amount of UNIX configuration information, organizes it, and displays it via a Web browser on request. Way cool!

We’re delighted to announce that Clif Flynn, author of Tcl/Tk for Real Programmers, has joined the ranks of ;login: columnists. To learn about Tcl and Tk from an expert, turn to “The Tclsh Spot.”

Investigating cryptography? Matt Curtin tells you when not to believe what the vendor tells you. Maybe you’ve been wondering about biometric authentication systems. Darío Forte has the latest word. Battling over ownership of root? Jim Hickstein takes a somewhat heretical stance: sysadmins should consider letting some users in on the ground floor. You’ve probably noticed that there’s more than one way to perform taxidermy on a feline, or to password-protect Web pages. As a cat-lover, I’ve censored the article on the former, but Dave Taylor gives the pros and cons of various approaches to the latter.

Spend 10 minutes, save 6 hours a day — of CPU time, that is. USENIX Board President Andrew Hume gives a real-life example of performance optimization. Glen McCluskey addresses the costs of data formatting. From other columns, learn how to write applets, how to help developers write apps your network can handle, how to analyze log files using Perl.

When you’ve taken advantage of all this information, you may start thinking about polishing up your resume. David Clark has some suggestions on how to make the most of your skills and experience.

Or you may be inspired to share your own hard-won techniques. If you’ve been working on the problems associated with UNIX on the desktop, please turn to Rik Farrow’s column this month. If your expertise lies elsewhere, remember, we’re always looking for ;login: articles, photographs, book reviews, apt cartoons, and outraged Letters to the Editor. Send email to jel@usenix.org.
letters to the editor

Correction: MRTG-2 Is Still Supported!
In the February 1998 login: there was a review of my LISA '98 MRTG talk. The article says that I am ending support for MRTG due to time constraints. This is not the case.

What I said in the talk was that I wanted to encourage people to migrate to rrdtool/cricket/mrtg-3 once these programs become stable. I am not actively continuing to develop mrtg-2 because I invest what time I have in mrtg-3/rrdtool.

But, as you can see from the mrtg-2 releases published in the last few weeks, I am still maintaining mrtg-2, although I am not adding new features but only integrating patches and removing bugs.

Cheers,
Tobi
Tobias Oetiker
<oetiker@ee.ethz.ch>

Correction: Solaris Certification Is Exam-Based
The column “Taming the Certification Beast” [October 1998 login:] made the statement that Solaris certification “relies on course attendance rather than testing.” This is inaccurate. The exams are based closely on Sun Educational Services course material, but the certification itself is based 100% on exam performance. There is no requirement whatsoever that a candidate for certification ever have attended a Sun course, and indeed there are a lot of Sun-certified sysadmins (and network admins) out there who did it all on their own.

I am not affiliated with Sun Microsystems in any way.

From the catapult of J.D. Baldwin
J.D. Baldwin
<baldwin@netcom.com>
We stand both cheered and chastened. – Managing Editor

The USENIX Crossword Puzzle

Across
1. Splendor
5. Landing gear support
10. Remain
14. Sword
15. Charity
16. Jason's ship
17. Repressor
19. Goat-tailed deity
20. Sets up
21. Lamented
23. Italian desserts
24. Racial epithets
25. Fast Canadian compiler
28. Two continents
31. Hunter in stars
32. Mustard plant
33. Yo ho ho drink
34. Encircle
35. Ringlets
36. Summon to court
37. Came into contact
38. Relating to liquid concentration
39. Business language
40. Slipshod
42. Tree group
43. Makes coins
44. Gasp
45. Nesty fly
47. Coast to the lower limit
51. Ninth Greek letter
52. Whale oil for candles
54. Two person combat
55. Brief
56. Andy's pal
57. Strays
58. Keaau movie
59. Finnish dude

Down
1. Gracious female
2. ___ Source Software
3. Army food hall
4. Cavil
5. Climber
6. Not heads
7. Executus
8. Employ
9. Unconditioned
10. African adventure
11. Record
12. Fit of shivering
13. Over there
14. Radar beacon (abbr)
15. Belonging to us
16. Aroma
17. Maggots
18. Uranus moon
19. Color measuring device
20. Matrix
21. Bars
22. Refine metal
23. Separates
24. Challenges
25. Involving a brain part
26. Abdo and speci follower
27. Cuban dance
28. Essential parts
29. Titled
30. Resolve to grammar parts
31. Periodic water movement
32. Tart
33. Absolute, undiminished
34. Cylindrical parasitic worm (short.)
35. Upon
36. At Language
37. Get up and go

Solution to the February Puzzle

REUSE BLOB AURA ALPHALINEUNIX IMPATIENCE DICE
SEEM CAKE AIMED ERR BETA ATOP
AID GAME EBB SCALD GENE IDEA AUGUSTA ENTREAT
GRIMANEW WIDTH SET EGG SMAS ALAS CARS FAR
ALTAR MAMA ALSO JOIN CAPITALISM ACNE ACES STEEP
RIGS BEDS HOSTS
I (Tina) recently took a look at the security implications of running SMB applications over the Internet. Initially, I thought the best place to look for information would be on an NT machine itself. I clicked on the Help button and proceeded to form the question. In particular, I asked about network services and bindings. The response to my first query looked promising:

binding:

To change the order of bindings for selected network components

To enable or disable binding paths for selected network components

To view bindings for network components

I selected the second one; and more help information appeared. The first part was a hand-holding walk-through on how to enable/disable bindings. Fair enough. But I really wanted to know exactly what services would break if I did so. I read on. The accompanying “Notes” section took the wind out of my sails in that regard, saying, “Do not attempt to change binding settings unless you are an experienced network administrator familiar with the requirements of your network software.”

And that was it! I poked around some more with the “Help” utility, but I couldn’t get anything more detailed than the warning. I began to feel frustrated, wondering how one was to “RTM” when the “M” isn’t there!

Sheepishly, but having at least tried to RTM, I turned to several NT gurus to ask them more about what services the bindings affect. To my surprise, many of them had the same questions I did. So I decided it was time to go sniffing for answers, and that you might be interested in what I found out.

Microsoft networking was originally designed for small networks. When Microsoft decided to extend it, they needed a capable transport protocol to do so. The result is NetBIOS over TCP/IP, or NBT. NT machines use ports 135 and 137-139 for all their Windows-related networking traffic. A breakdown of the ports looks like this:

135 loc-srv Location Service
137 netbios-ns NetBIOS Name Service
138 netbios-dgm NetBIOS Datagram Service
139 netbios-ssn NetBIOS Session Service

Location Service is like the UNIX portmapper and is used to get information about the RPC programs registered on the machine. NetBIOS Name Service is used for registering and gaining information about NetBIOS names. The NetBIOS Datagram and Session Services can be viewed as UDP and TCP for NetBIOS packets.

I wanted to turn off selected Microsoft networking services so that an NT machine wouldn’t be running file and printer sharing (a.k.a. Server Service) portion of the SMB applications on the...
Internet, but could still run a Web server, for instance. In NT you create “bindings” between logical connections and services, protocols, and adapters. I hypothesized that turning off pieces of NetBIOS on the Internet-connected adapter would disallow the targeted SMB services and still allow more traditional Internet services. I used the Network property sheet from the Control Panel, which contains the Bindings tab, to toggle bindings on and off. While I systematically turned on and off the services on the NT machine, I used SAMBA commands from a UNIX machine to look at the NT responses.

First, I turned off the Control Panel-level Server Service. As expected, the NT machine didn’t answer a NetBIOS name query at all:

```bash
% nmblookup -B 204.146.133.23 -S \*
Sending queries to 204.146.133.23
name_query failed to find name
```

I reenabled the Server Service and then, using CONTROL PANEL/NETWORK/BINDINGS [all adapters], turned off WINS Client [TCP/IP]. Again, as expected, the NT machine failed to respond to a WINS name query:

```bash
% nmblookup -B 10.31.3.163 -S \*
Sending queries to 10.31.3.163
name_query failed to find name
```

Next, I reenabled the WINS Client [TCP/IP] and tested to make sure that the machine was responding. In the successful NetBIOS name query, we see that the machine lists a NetBIOS name type of <20>, which indicates a resource-sharing "server service"; this is what we would expect. Since we have a server service, we can now use a subsequent smbclient query to that server.

```bash
% nmblookup -B 10.31.3.163 -S \*
Sending queries to 10.31.3.163
10.31.3.163 <00>
Looking up status of 10.31.3.163
received 10 names
PI <00> - B <ACTIVE>
```

**INet-Services**
- <lc> - <GROUP> B <ACTIVE>
- PI <20> - B <ACTIVE>
- IS-PI <00> - B <ACTIVE>
- SUNNYVALE <00> - <GROUP> B <ACTIVE>
- PI <03> - B <ACTIVE>
- SUNNYVALE <1e> - <GROUP> B <ACTIVE>
- ADMINISTRATOR <03> - B <ACTIVE>
- SUNNYVALE <1d> - B <ACTIVE>

```
__MSBROWSE__

num_good_sends=0 num_good_receives=0
% smbclient -L PI -I 10.31.3.163
Added interface ip=10.31.3.161 bcast=10.31.3.255
nmask=255.255.255.0
Server time is Wed Jan 27 14:32:07 1999
Timezone is UTC-8.0
Password:
Domain=[SUNNYVALE] OS=[Windows NT 4.0]
     Server=[NT LAN Manager 4.0]
security=user
```

This machine has a browse list:

```
Server Comment
----- -----
PI
```

This machine has a workgroup list:

```
Workgroup Master
----- -----
SUNNYVALE PI
```

A machine configured in this way would serve File and Print shares on the Internet, which is the NetBIOS service I ultimately wanted to turn off.

Finally, I expanded the “+” WINS Client [TCP/IP]. It shows three bindings that can be toggled on and off: NetBIOS Interface, Server, and Workstation. With the top-level WINS Client [TCP/IP] binding still enabled, I disabled the “Server” binding (below WINS Client [TCP/IP]). Note that we can still successfully query for the name, but the results show there is no longer a server service, type <20>, on the machine. As a result,
the subsequent smbclient query is unsuccessful and requests for File and Print shares would fail.

% nmlookup -B 10.31.3.163 -S \
Sending queries to 10.31.3.163
10.31.3.163 *<00>
Looking up status of 10.31.3.163
received 5 names
PI <00> - B <ACTIVE>
INet-Services <1c> - <GROUP> B <ACTIVE>
IS-FI <00> - B <ACTIVE>
SUNNYVALE <00> - <GROUP> B <ACTIVE>
PI <03> - B <ACTIVE>
num_good_sends=0 num_good_receives=0
% smbclient -L PI -I 10.31.3.163
Added interface ip=10.31.3.161 bcast=10.31.3.255
rmmask=255.255.255.0
Session request failed (131,130) with
myname=EPILOIU destname=PI
Called name not present
Try to connect to another name (instead of PI)
You may find the -I option useful for this

Don’t let the lack of detailed online NT documentation dissuade you, or the counterintuitive naming of the WINS “Client” [TCP/IP] binding fool you. Expand the “+” on the WINS Client [TCP/IP] binding to see that underneath the “Client” lies a “Server,” which you can enable/disable. It might help to think of the WINS “Client” as NBT, and the bindings underneath as pieces of NetBIOS over TCP/IP. The bottom line is that there is a degree of granular configuration control for NBT using bindings to disable all or pieces of NetBIOS on a single adapter.

References


Broken Paradigm

by Hal Miller

Hal Miller is president of the SAGE STG Executive Committee.
<halm@usenix.org>

Today I placed an order for over $5 million in computing equipment. There will be many times that to come — certainly the biggest single project I’ve ever worked on. What did I buy? Mostly storage: 8 terabytes. At the current growth predictions (much of which is already funded), I will approach, if not exceed, a petabyte in the next four years.

“Neat!” you say. So did I. Then I realized: “bandwidth between disk and servers.” Then I thought, “Oh no! Backups!”

Technology continues to advance. So does the demand for it. There is a significant gap between those rates of growth, and the future looks difficult for those of us tasked with using the former to supply the latter. Let’s look at my real-life situation as an example, then see what we might do about at least breaking the problem down into solvable chunks, if not solve it as a whole.

With that much disk online, in a heavy-use environment (read-write all over, 7 x 24 x 52, lots of users, 90-day-long jobs), getting data back and forth between storage media and CPU servers is a problem. With that many heads and spindles to manage (over 1000!), the seek time for a given bit of information can be long. Given that it’s all random-access filesystems (well, there is a database, too, just for complications, but it’s “relatively” small), there isn’t much of a way to index around and cut down search time. This is all UNIX filesystem. We have known for years that the UFS is nearing an upper limit on directory size, and it appears to have other limits as well.

How do we deal with data integrity? There are RAID5 and mirroring solutions, among others. Who wants to pay for the extra disk (let alone computer room space, power, and air conditioning) for my mirror?

Storage Area Networking is a solution for some of the bandwidth issue. But, as with the other points, how long, at this rate, before we outgrow that? Probably just about the time we finish our first backup.

Speaking of which, the backup paradigm we all know dictates copying either blocks or files to tape, in some pattern to allow for restoral of data after hardware failures in some “reasonable” amount of time (plus, in some places, to allow for restoral data after user error). I’m putting 16 DL17000s into this. Filling the tape library cabinet costs nearly $50,000 retail and will cover a week or so. Filling the tapes with data may take more than that
week. That means I need to change what "backup" means. I can't take anything offline to dump, so I need either to "break" a mirror or back up a "snapshot." What technology I apply is really not the issue here (nor am I looking for those other large sites out there to pick on my scenario) - whatever that technology is, we have already outgrown it, or will soon.

Enough on disk and backup. How about security? My site is pretty well hidden. There isn't a lot of reason for people to come looking for us except that our router answers up on the Net like everyone else's. We are under scan or more concerted attack every few hours, perhaps more; I have no control over my current network and can't really see effectively. Fighting this, recovering from those incidents we've had (Linux mounted and NT, all boxes I didn't know were brought in and connected), is a full-time job, and I don't have anyone to apply to it. Tool building for IDS and other parts of the game proceeds, but not fast enough.

Technology advances have been staving off "defeat" for a while and will continue to attempt that, but we as an industry are losing the battle. Demand continues to skyrocket. Paradigms are stretched to the point of breaking throughout the computing world.

So what do we in SAGE do? Hard question without obvious answers. Let's start with what we can't do, and see what's left. Then, remembering our job as sysadmins seems to include "performing magic" to solve whatever odd problems nobody else dealt with, we will try to pull yet another rabbit from the hat.

We can't develop better hardware solutions. We can't fund new hardware products. Most of us aren't advanced hardware engineers. Maybe the vendors can do these things, but they aren't likely to do so of their own volition, since they make good money selling what they have to offer us now. We need to apply our reputation and efforts to convincing vendors to join us in a "consortium" type of effort to devise new long-term solutions. We can sponsor workshops calling for work in progress, brainstorming sessions, or joint work proposals. We can fund our own members to work on software tools if they will return benefit to our community. We can put our collective experience together into reviewing what the requirements really are and designing methods to meet them. We can get vendors to build it if we show them what we want.

This year I would like to see the formation of a SAGE Development Fund, and a SAGE Vendor Liaison function. I hope they make some progress before I add the next few dozen terabytes to my backup system.

The results of the election for the seven

![Election Results](Image)

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

SAGE Executive Committee positions of the USENIX Association for the 1999-2000 term are as follows:

- Barbara L. Dijker 542
- Hal Miller 520
- Peg Schafer 431
- Timothy Gassaway 426
- Xev Gittler 411
- Jim Hickstein 362
- Geoff Halprin 353

Not elected:
- Bruce Alan Wynn 309
- David Parter 289
- Bryan MacDonald 257

Total number of ballots mailed: 4,337
Total number of ballots cast: 656
Return rate: 15%
Total number abstained: 2
Total invalid ballots: 1

Newly elected SAGE Executive Committee members took office and chose their own officers at their executive committee meeting held 22 February 1999, in New Orleans. The new Executive Committee officers are:

President: Hal Miller
Vice President: Barb Dijker
Secretary: Tim Gassaway
Treasurer: Peg Schafer

SAGE Certification Subcommittee Briefs

Dan York has been added to the subcommittee. Dan is a technical instructor and training manager and is a volunteer for the Linux Institute, a community project established to develop professional certification for Linux.

[Link to Linux Institute website]

The Human Resources Research Organization (HumRRO) has been selected to conduct comprehensive research on system administration as an occupation and perform an occupational analysis.

The research will include review of existing materials and data as well as active measures such as surveys and focus groups. This phase of the project should be completed in August.

SAGE is also seeking sponsorship from individuals or organizations who wish to contribute to the certification efforts.

See our Web site for more information.

[Link to SAGE certification page]
I was struck with a bit of vertigo recently. Reading through the recent missives about the virtues and vices of certification was the culprit.

The SAGE certification committee has been trying to get the topic out on the table so that all the views can be aired. There have been articles in jlogin. At LISA there was a debate. Since then, the sages-members mailing list has had a revived discussion. Finally, comments and pleas are being sent to the SAGE certification advisory council, and discussion has been brewing there.

And there it was. Dizziness, nausea. The cause is not that I’m sick of it the topic. The cause is that the discussion has not only come full circle, it is going in circles over and over again. There may be a few new arguments for certification related to changing markets. However, not a single new insight has been raised against certification in six years.

At the very heart of the debate, and expressed incessantly in an infinite number of creative ways, is the question of whether what a system administrator does can be put in a tiny little box (cubicle, box, cubicle, box) and neatly labeled, categorized, and evaluated. Gut reaction to that idea is pure repulsion. It threatens our identity. It undermines all the hard work we’ve done to get this far.

Most system administrators practicing today have formal training (if any) in a vaguely related area and learned everything the hard way. My education is in physics and I used to program spacecrafts. What brought most of us into the “profession” was a drive to figure things out, the ability to learn things that weren’t well documented, and the naivete to think others would be grateful. How do we measure that? Few of us have had any formal education in system administration – because it didn’t exist. Due to our experiences, we find it difficult to consider that one can really learn system administration any other way.

At the same time, we all complain about being overworked. I personally can’t wait for genetic cloning. Then I (and my six other clones) will be able to work something less than a 12-hour day and have a “normal” evening at home. There are now, after about 10 years, a good number of excellent tutorials, classes, and books on various aspects of system administration. We seem to be able to impart our knowledge base. We should be able to evaluate one’s application of that knowledge. We should be able to certify that evaluation. There appears to be value in doing that.

Every “profession” goes through a maturation process. I have no doubt the first doctors were just as arrogant that their skills could not be duplicated or measured. Some still are. Same with lawyers or any other group doing work which involves significant experience and cognitive process. But as the work evolves, the knowledge and practice of the work evolve and can more readily be imparted and evaluated. Think about where we might be without certification of existing professions like nursing. Certification is never a replacement for apprenticeship or on-the-job experience, it’s a foundation upon which to build.

Maybe system administration isn’t ready for certification yet. But one day it will be. How will we know when? Probably when the first-generation system administrators are long gone and the new generation can’t remember their old arguments against it. The new generation will become system administrators not through the School of Hard Knocks but through one of any number of training programs already available, which are improving all the time.

The 1998 Outstanding Achievement Award

“Presented to Tina Darmohray for her dedication and tireless efforts which promote understanding and recognition of the System Administration Profession”
I joined the corporate server support group for SAIC as a senior UNIX system administrator in June of 1998. Immediately I began to get lots of questions about the configuration of various UNIX boxes that the group had on hand:

"How much memory?"
"How much disk storage?"
"How are the file systems laid out?"
"Is it running NIS?"

I began to profile a number of these systems by creating a template in an MS Word document that described the general identity, hardware, network, and software configuration of each of these systems. Up to seven or eight systems, this arrangement worked just fine. However, we kept getting involved with more systems (or people saw our profiles and asked us to do ones for them). I'd also think of another piece of information to add to the profile and would have to go through and edit each of the profiles to keep them in sync. All in all, the unautomated task of maintaining these profiles was becoming unwieldy.

I began looking around for existing tools for automatically profiling UNIX systems (especially useful when you have 40 or 50 servers from several vendors, with varying configurations, rather than hundreds of identical workstations), but I found very little existing software. Russ Allbcry pointed me to one C program, SysInfo, but it seemed to deal mostly with tunable parameters.

Ultimately, I wrote a series of Perl scripts to try to automate the profiling process, mostly as a "proof of concept." The requirements were:

- Collect information using Perl 5 or later.
- Don't require any additional CPAN modules, if possible.
- Avoid exotic "deep magic" system calls, if possible.
- Allow certain fields to be filled in by the sysadmin.
- Allow these sysadmin fields to "include" multiple external files (in the same directory), including text and graphic files.
- Allow the fields collected to be extended.
- Be aware of security concerns, since some of the information can only be collected if you are running as root.
- Allow transmission of the collected information by email (so the process can be automated with cron).
- The system that receives the messages needs to able to have an "agent" process these incoming messages without human intervention.
- Allow the profiles to be displayed by Web browser.
- The Web server displaying the profiles needs to able to support Perl CGI scripts.
- Allow the final format of the profile to be determined by a format file rather than be hard-coded.

As of this writing, these scripts work for HP-UX 9.X and 10.X and for Linux systems. By the time you read this, I hope to have extended them to Solaris 2.5 and 2.6 systems, and perhaps to Windows NT. One set of scripts automatically generates profile information.
Since I'm probably not the only person to get these kinds of questions, I am wondering if there is any interest within the SAGE community to collaborate on this software to extend it, refine it, port it to other "species" of UNIX, and make it available to the SAGE community as a whole. I've gotten the permission of my manager to release the Perl source if there's sufficient interest. 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.mahler@saic.com>

The profile information is emailed to another system where a Perl script handles the incoming message. If it's the first profile for this system, then a subdirectory is created and the profile is stored there; otherwise it is merged in with existing profile information for that system. A third Perl script serves as a CGI script to generate a form to query for system profiles, summarize all of the profiles that exist, and create the HTML for the profile of a requested system. The code that generates the profile output is driven by a format file so that a local site can customize the format of its Web-based system profiles without having to hack a single Perl script. This entire process, while perhaps not elegant, works end-to-end today. The rest of this article provides more detail about the software, which I call **syssumm**, as it exists. It is taken from the existing README file and includes some examples of the files and Web pages generated.

### Components of **syssumm**

The **syssumm** software has two parts: the software that runs on the remote systems and collects the configuration information, and the software that runs on the Web server, processing incoming messages and merging the information into the appropriate subdirectories.

The "remote" software is composed of a Perl "driver" script called **syssumm.pl** that makes calls into OS-specific Perl modules, plus a common Perl module of utility subroutines. In general, there is a Perl module for each vendor's OS. For example, for Solaris, HP-UX, and Linux you would have `SunOS.pm`, `HPUX.pm`, and `LINUX.pm`

Only the module for the local OS running on the system is "used" by the **syssumm.pl** script. For example, if you are running the script on a Sun box, the `HPUX.pm` and `LINUX.pm` modules are never loaded. When it came to extracting information about the remote system, **keeping things simple** was valued over being extremely clever.

The "Web server" software is composed of a Perl script called **incoming.pl** that extracts and processes incoming email messages, and a CGI Perl script called **sysquery.pl** that generates an HTML form, then processes the returned results of that form and displays a system summary. The CGI script also generates a Web page summarizing all existing system profiles.

### Software Requirements

On the remote systems the only requirement is Perl 5.x. No additional CPAN Perl modules are required to run the profiling script. On the Web-server system the only requirement is a sendmail-like MDA that allows the processing of messages through a .forward mechanism and Perl 5.x. The Web server must support handling CGI scripts written in Perl.

### How Does the Software Work?

First, the system administrator sets up the Web-server software, creating a new account on the system where the Web server exists with a userid of "syssumm". The output of **syssumm.pl** would be emailed to this account.

Next, in the home directory of the "syssumm" account, the system administrator creates a .forward file which delivers each incoming message to the **incoming.pl** script that processes the messages and places them in the appropriate directory under the Web server.

The system administrator loads the "remote" software onto a system to be profiled and either manually runs the **syssumm.pl** script or sets up cron to run periodically. The "-m" command-line option specifies the email address to which to mail the output. (This should be "syssumm@Webservice.yada.yada.yada")
Next, a pointy-haired user brings up the form (generated by `sysquery.pl`) and enters the name of a system, which is sent back to the same `sysquery.pl` script to interpret. If the name equates to an existing system for which there is profile information, then that information is formatted as HTML and returned to the requestor's browser. Figure 1 shows what the query form looks like.

A format file is used to control the appearance of the Web page so that you can choose what fields to display in your system summaries and in what order they appear. Figure 2 shows what a system profile would look like.

The existing `sysquery.pl` script is intended to be an example of how to query and retrieve system profiles; obviously, your local Web pages will have a different look.

**Appearance of the Generated Data**

The information generated by `syssumm.pl` and processed by `incoming.pl` is a basic ASCII flat file. Each line is self-sufficient and is composed of:

```
  category:sub-category:value
```

The following categories have been "hard-coded" into the script that creates the output:

- General
- Hardware
- Network
- Software
- Comment

Within each category are subcategories. For example, within the General category, you'll find:

- NodeName
- Organization
- Vendor
- Model
- ...

Some of the subcategories are "indexed." For example, the subcategories for disks and tapes look like:

- Disks-0
- Disks-1
- Disks-2
- ...
- Tapes-0
- Tapes-1
- ...

Each line represents an individual device.

Entries in the Comment category are optional. They would be remarks such as "Run as root to get more information."
This is what the actual lines would look like for a hypothetical system:

General:nodeName:ornithomimus
General:Organization:PROTECTED
General:Vendor:Sun
General:Model:Geewhiz
General:HostId:12345678
...

Note that certain lines contain the value "PROTECTED". These fields signify that this information cannot really be figured out by a stupid Perl script and needs the omniscience of a human system administrator.

If a system has been profiled in the past and an updated profile is sent to the Web server system, PROTECTED fields will not overwrite prior contents. This is especially effective when the system administrator has gone in and provided the information that the Perl script couldn’t figure out. This feature protects the work that they’ve done so that they don’t need to fill in those fields again.

Note that when the incoming.pl script processes PROTECTED fields (especially for the first time), it changes "PROTECTED" to "To be provided" (to make the report more readable).

**Fields Collected by the Profiling Scripts**
The following list summarizes the categories and subcategories collected.

**General**
- NodeName
- Organization
- Vendor
- Model
- HostId
- SystemHandle (PROTECTED)
- Location (PROTECTED)
- DateSystemInstalled (PROTECTED)
- LargerPicture (PROTECTED)

**Hardware**
- Processors
- Memory
- Disks (Indexed)
- Tapes (Indexed)
- Console (PROTECTED)
- OtherPeripherals (PROTECTED)

**Network**
- DomainName
- DefaultRouter
- NameServer
- NetworkInterfaces (Indexed)
- HardwareNetworkAddress

Vol. 24, No. 2
Software

OsName
OsVersion
NbrOfLicenses
Patches (Indexed)
RunningLpsched
Printers
RunningSendmail
RunningNfs
RunningAutomount
RunningNis
RunningNameserver
RunningSamba
GraphicalUserInterface
FontServer
LocalFileSystems
RemoteFileSystems
RunningAccounting
ApplicationsInstalled (PROTECTED)
InstallationProcedure (PROTECTED)
DataFlowsTo (PROTECTED)
DataFlowsFrom (PROTECTED)

Comments

The “LargerPicture,” “DataFlowTo,” and “DataFlowFrom” PROTECTED fields are intended to provide the context that the system resides in within your local data center.

Note that while items such as patches are summarized, tunable parameters are not. This may be an area of interest for some system administrators to profile.

Multi-line Values and Extensibility

The system is extensible in the sense that you can add fields to the profile. When incoming.pl merges a new profile into an old one, non-PROTECTED fields “over-write” existing fields, existing PROTECTED fields are unchanged, and new PROTECTED fields are installed in the profile as “To be provided.” As long as you add them to the format file that controls the CGI/HTML output, they will appear in any system profiles requested after that point.

PROTECTED fields are allowed to point to one or more files that may contain text or (Web-compatible) graphics. In fact, if the value begins with a “.” (period) or “/” (forward slash) — indicating a relative or an absolute path, respectively — and if that file exists, then it will include the contents of the file in place of the value.

Summary

These scripts are the humble beginnings of a system that can profile general, hardware, network, and software configuration information about UNIX systems and display the output on any Web browser.
resume your resume writing

Of all the stumbling blocks for techies in search of a career change, writing a resume is high on the list of possible points of procrastination. Fortunately, unlike many software professionals, I've always enjoyed writing and formatting resumes; it presents a challenge to me, and I enjoy the results. Over the past seven years as president of a technical-staffing company, I've written many thousands of resumes, most of them for UNIX system administrators. I'll share a few tips and some useful anecdotes that may make writing yours a more palatable, and profitable, proposition.

The primary purpose of a resume is to land an interview. However, the resume, or curriculum vitae (CV), is a part of an overall marketing plan that will assist you in re-navigating the waters as your job demands. Resumes are also maintained by human-resources (HR) staff as a credential for your claimed abilities after you are hired. Resumes of prominent individuals in corporations are often provided for contractual purposes such as engaging in a service-level agreement. You'll need a resume for many purposes as you proceed along your career path, so make it a good one.

Writing a resume is pretty straightforward. First, figure out who your target audience is and what they need to know, and then aim it towards them. For most of us, the target audience is someone who can bring us in for an interview. Avoid HR people and resume mills as your target interview audience, unless you are desperate. Instead, I advise creating resumes for hiring managers and finding ways to get your resume directly into their hands.

In my mind, HR stands for Huge Roadblock. The following anecdote backs this up. A friend of mine, who was as a manager at a large database vendor, was discouraged because she wasn't getting any candidates for her open positions. Another friend of mine was looking for work as a database administrator and was a perfect fit for the position. It turned out that he had sent his resume to her company almost a dozen times in the past year, mentioning the applicable job numbers in his cover letters, with no response. When he was referred through word of mouth, the hiring manager screened and hired him on the spot. HR people are generally capable of producing swank holiday parties and finding a dentist in your HMO group, but don't count on them to help you find a job. So, if possible, always aim for the hiring manager.

System administrators should focus their resumes on two criteria: first, whether you are looking for contract or full-time permanent employment and second, your level of seniority. Resumes are typically no longer than two pages. Try to focus what you really need to say in that amount of space.

Cover Letters and Packaging

Make your presentation clean and to the point. Include a very brief cover letter. Don't create a point of objection by mentioning your gun collection or religious beliefs. The cover letter is only an introduction; it will probably get tossed or stapled to the back of the resume.
Greetings,

In response to your ad in the Washington Post for a Senior UNIX Analyst I am including my resume for your consideration. I may be reached after hours at my home number or via email to my personal account.

<at this point, if you have a brief item that warrants special attention, set it out as a hook, but don't give too much away>

I have worked at your competitor, Xylcorp, for three years and was responsible for architecting their entire facility.

Thank you,

<sign here>

Remember that your resume will probably be “processed”; try to anticipate the clerical person on the receiving end and avoid folding the resume more than twice. For another 32 cents you can send the resume in a 9x12 envelope with no folds; this stands out.

**Distribution**

I've heard of people sending over 500 resumes out at a time in a mass marketing effort. This is a great big mistake. People who receive resumes don't want to feel that you have disseminated the document widely. In fact, the hiring staff wants the impression that this is a special overture for them alone. Instead, plan your career moves, select specific companies that interest you and find ways to get the attention of the right people inside that company. Do a little shopping while you are still happily employed; peruse the want ads or newsgroups. Poke around. You're better being selective in your job hunt. It's a better use of your time and everyone else's.

**Format**

It's always easiest to start from an example, so I'll include one of mine. Historically, resumes were sent in a #10 envelope and folded twice into three sections. Because of this, the top third of the resume is considered the place to catch the reader's attention.

I'd avoid using an "objectives" header. It can be too limiting and provide a good excuse not to read the rest of the resume. Throughout, use specifics, depth, and consistency to catch the eye of the hiring manager. For a particular position, I start with the job requirements and follow them as an outline, insuring I've addressed as many applicable points as possible.

**Dave's Resume Format**

NAME

CONTACT INFO

CRITICAL SKILLS

A complete paragraph that concisely explains your abilities and interests. This section can easily be changed to tailor the resume to specific jobs. Move the most relevant material to the first few sentences.

>> fewer than ten sentences <<

Highlights, optional acronyms, buzz words supporting critical skills, e.g.,

System administrators should focus their resumes on two criteria: first, whether you are looking for contract or full-time permanent employment and second, your level of seniority.
Weave a thread. Show that the previous job led to your taking on new and different responsibilities.

OS: Solaris 2.6, HP/UX 9.x, Irix 6, Linux, FreeBSD, NT
Hardware: Sparc, board level and FRUs
Scripting: Perl 5.x, Ksh, light TCL
Facilities: DNS, Sendmail, RADIUS

>> five lines max <<

EDUCATION

College degree and recent germaine technical training

>> three lines max; don't list your graduation date <<

PROFESSIONAL WORK EXPERIENCE

>> HEADING <<

Company_Name (6 months)
Senior Unix System Administrator. Responsible for ...

OR

Company_Name
Sept '98 to present
Operations Engineer. Provided back-line support for ...

>> Use parallel construction. <<

There are three or four forms of listing job experience, just use one of them consistently.

>> NEVER refer to yourself in the third person, e.g., "Mr. Jones created a backup system to handle." <<

>> Unless you are in a management role, it's generally advisable not to toot your horn about cutting costs, increasing efficiency by a multiple, etc. <<

>> Weave a thread. Show that the previous job led to your taking on new and different responsibilities. <<

>> Show lots of details. Managers love specifics, e.g.,

WRONG: "Worked at very big site with big, cool computers."
CORRECT: "Served in a team which managed a 7x24 site with over 500 Solaris 2.6 desktop systems and 24 servers."

WRONG: "Wrote backup systems."
CORRECT: "Modified existing Perl 4.x backup systems to interface Legato version 2 software with HP, and Solaris file systems totaling 700 gigabytes."

>> Put the most bulk in the most recent job. For sysadmins I can usually fit the two most recent jobs on the front page together with critical skills and education. <<

>> Older jobs should be pared down to shorter paragraphs. If the job is over 10 years old, consider leaving it off or doing an honorable mention with a modest sentence. No one really cares about how many lines of CPM assembler you wrote in 1982. <<
OTHER TRAINING, CERTIFICATIONS, AND ASSOCIATIONS

Noncollege work
Historical training classes
USENIX, IEEE Membership

COURSEWORK

Avoid listing coursework on a resume unless this is your first job out of school. Refer only to coursework that is specific to the job field you are applying for, e.g., “Thesis: 3D rendering algorithms for multi-tasking operating systems.”

HOBBIES / PERSONAL INTERESTS

National Rifle Association
US Postal Workers Support Group
Tai-Kwak-Wo Kick Boxing Champions of Gilroy
Abott Labs Pharmaceutical Research Volunteer

>> Consider what you want to tell the reader in this section of your resume. If you must show personal information, limit it. <<

If you have been out of school for more than five years, avoid this section. Hobbies and personal interests do not belong on the resume of a consultant.

Tips and Observations

If you are surface-mailing your resume, avoid using extra-fancy paper stocks; most of the time it will be photocopied. Use a serif font between 11 and 14 point. Unless you are an inventor or marketing whiz, avoid gimmicks like rubber gloves, petuli oil, pictures, fancy images, unusual designs, or excessive colors. Leave a sufficient amount of white space and borders.

Unless you speak French, don’t put the acute accent marks over the word “resume,” even if you did have to write a PostScript subroutine to make it perfect. It’s an affectation and a distraction.

Be warned that an emailed resume can be altered easily. Ditto the resumes posted via URL. Don’t send your references until you have talked to a human, there is a strong interest in that company, and that company has explicitly asked for them.

Finally, be particular about quality details, spelling, formatting, copy quality, and legibility.
bandwidth versus latency

Helping Developers Understand Network Performance Limitations

Software developers and integrators frequently don’t understand what performance they can reasonably expect from a corporate network. This often leads them to design and implement software that works great on a LAN but is utterly unusable when deployed across a WAN. To address this at our company, I put together the following short email message to our developers, explaining the basics of network performance (particularly the difference between bandwidth and latency), describing how this affects client-server applications, outlining what performance they should expect from our corporate network, and suggesting guidelines for their client-server development efforts. If you face the same kind of constraints in your business, you might consider providing something similar (with numbers changed appropriately) to help your own developers understand their environment better. You can also apply this same principle in other areas, such as CPU, memory, or disk performance, to help folks understand what they can expect.

I'd like to suggest that any application or service that you're designing to run over our corporate WAN should be designed so that it performs adequately under the following conditions:

- Single user
- Single task (no other applications using the WAN link simultaneously)
- 128 kb/s bandwidth
- 100 ms latency (one way)

We will have much more bandwidth than that available to most corporate sites (but not all sites, and not all the time). We should use that bandwidth, however, to accommodate multiple simultaneous sessions (from multiple users, from multitasking individuals, or from some combination of both), rather than counting on it to provide adequate performance for any individual session.

If you keep these guidelines in mind as you develop applications and services, it will go a long way toward ensuring that our applications and services work well across our entire corporate WAN.

I'm not suggesting these numbers arbitrarily. There is some logic behind them: 128 kb/s bandwidth is what you can expect out of an ISDN line. Right now, for cost reasons, all our corporate regional offices are limited to 128 kb/s burst speed for bandwidth, on a public-carrier frame relay WAN. Each office should have more bandwidth available in a few months when [something confidential] happens, but they'll still have backup connectivity that's limited to 128 kb/s (either their current frame relay service, or dial-on-demand ISDN service). And they'll still have folks who want to use
the apps from home, some of whom will be limited to only ISDN bandwidth (128 kb/s).

As for latency, 100 ms one-way is about what we can expect out of our coast-to-coast frame relay–based IP WAN during peak usage periods each day. One-way latency on our LAN at headquarters is usually 3-5 ms or less. A task where a client/server application or service does 10 round-trip conversations between the client and the server on a path with 5 ms one-way latency can still complete in 100 ms (1/10th of a second), which “feels fast”; that same task on a link with 100 ms latency will take 2000 ms (2 seconds), which almost certainly “feels slow.” . . .

The upshot of this for software developers is:

- Don’t develop applications or services that need to transfer data between the client and server at more than 128 kb/s in order to avoid “feeling slow.”
- Don’t develop applications or services that require lots of little back-and-forth messages between the client and server; you’ll get eaten alive by the latency.

Remember to include whatever numbers are appropriate for your site and to clearly outline the performance constraints and expectations plus the technical explanation for them. Since it’s easier for everyone to accommodate design specifications up front, rather than retrofit applications after the fact, everyone will appreciate this “heads up” information.

Help your own developers understand their environment better. You can also apply this same principle in other areas, such as CPU, memory, or disk performance, to help folks understand what they can expect.
Perl is particularly well-suited for the analysis of log files and other similarly organized text. Using Perl, you can search files for entries meeting particular requirements (as with the `grep` command, but more powerfully), you can build data structures that capture and organize the contents of files, and you can summarize or restructure data in the files. In this column, I'll illustrate the techniques that Perl programmers use to perform these tasks, starting with the basics, then proceeding to more sophisticated examples.

**Processing One Line at a Time**

Many log files are organized so that each line is a separate "record" in the log. Generally, you want to process this type of file one line at a time. The idioms for this in Perl are the ubiquitous:

```perl
open FILEHANDLE, '/my/file' or die 'couldn’t open: $!';
while (<FILEHANDLE>) {
    # do something with the contents of $
}
close (FILEHANDLE);
```

The `while (<FILEHANDLE>)` loop is a shorthand way of writing:

```perl
while (defined($_ = <FILEHANDLE>)) {
    # do something with the contents of $
}
```

Both these snippets read a line at a time into `$_` from the file opened as `FILEHANDLE`. Inside the `while` loop, you put whatever code is necessary to process a line of the file. For example, to print all the lines containing the word `5esigma`, you could write:

```perl
while (<FILEHANDLE>) {
    print if /\b5esigma\b/;  # print and // both default to $
}
```

You might choose to extract information during the loop and then print it out in some other form after the file has been completely read. Often, you will want to read data into a hash as part of this process. For example, to parse the `passwd` file and create hashes that map user names to user ids and vice versa — a bit of makework, mind you, because this capability already exists in the built-in `getpwnam` and `getpwuid` operators — you might write:

```perl
open PASSWD, '/etc/passwd' or die 'couldn’t open passwd: $!';
while (<PASSWD>) {
    chop;
    my ($name, $dummy, $uid) = split /:/;  # split defaults to $
    $uid($name) = $uid;  # add a new name/uid to $uid
    $name($uid) = $name;  # add a new name/uid to $name
}
close (PASSWD);
for (sort keys %uid) { print 'uid for $_ is $uid($_)\n' }
for (sort { $a <=> $b } keys %name)
    { print "name for $_ is $name($_)\n" }
```

Note that I am spelling `foreach` as `for here`. The `foreach` and `for tokens are interchangeable.
The split operator breaks each line of the password file into its constituent fields. We assign the first and third fields to $name and $uid, respectively, then use those values to create hashes. (Note that there is no conflict between the scalar variables $name and $uid and the hashes $name and $uid – they are independent.) The last two lines print out the contents of the two hashes. Because the keys of $name are numeric user ids, they must be sorted in numeric order rather than the default "ASCIIbetical" (character-by-character) order; thus the sort block ($a <=> $b).

**Reading Multi-Line Records**

You may occasionally encounter text files where records occupy several lines and are set off from one another by delimiting lines. Perl's scalar .. operator, also known as the “flop” operator, is sometimes helpful in dealing with this type of file. Suppose, for example, that you are parsing a file consisting of records that look like the following:

```perl
begin user joeblooe
name: Joseph N. Hall
phone: 555-1212
email: joseph@5sigma.com
end user
```

The following code will scan input one line at a time and print out only the record(s) for the user joeblooe:

```perl
while (<>) {  # read from standard input or files in @ARGV
    print if /^begin\s+user\s+"joeblooe\"/ .. /^end\s+user/;
}
```

The flop operator works by maintaining a “state” that is either true or false. Each flop operator in a program has its own state. The flop operator starts out yielding false, and first yields true when the lefthand expression evaluates to true. It then yields true until the righthand expression evaluates to false. It's a slightly obscure feature of Perl, but, as you can see, when it's right for the job it can yield very succinct programs.

**Reading a File All at Once**

Perl programmers tend to read files one line at a time – Perl has a lot of features that work well on “line at a time” input, and if lines have a known maximum length, you can be assured that a program reading one line at a time can handle a file of any length. However, sometimes you may want to read the entire contents of a file all at once – to do some multi-line pattern matching, or for efficiency, or “just because.” The customary way to read all of a file is to clear the line separator variable $". If $" has the value undef, the line input operator <> will read the entire contents of input into a $scalar rather than a single line from it. Here is an example where we read the password file all at once and create a hash of the names and user ids in one fell swoop:

```perl
{  open PASSWD, '/etc/passwd' or die "couldn’t open passwd: $!";
    my $oldfh = select PASSWD;
    local $/;  # undef $/ for PASSWD in this block
    select $oldfh;  # restore previous default filehandle
    $uid = <$PASSWD> =~ /^(.*?):\1:.*$/;  # all at once!
}
for (sort keys $uid) {  print ‘uid for $_ is $uid($_)\n’ }
```

There is a different $/ for each filehandle. In this example we have to use the select operator to make PASSWD the current filehandle so that we can change the value of its $/. Next, we restore the previous current filehandle (probably STDIN), then read the entire contents of the password file and perform a match that returns a list of name and user ids suitable for initializing a hash – note the /m and /g options in the match operator.
Searching Simultaneously for Multiple Patterns

Sometimes you will want to search a file for lines matching one of several patterns. Certainly, you could write something like:

```perl
while (<FILEHANDLE>) {
    print if /joeph\b/i or /hall\b/i;
}
```

You can interpolate variables into match operators if you want to specify patterns at runtime:

```perl
($pat1, $pat2) =qw( (?i)\bjoseph\b (?i)\bhall\b); while (<FILEHANDLE>) {
    print if /$pat1/ or /$pat2/;  # (?i) gives case-insensitivity
}
```

You have to be concerned about a couple of things when interpolating variables into match operators. First, the variables must contain legal regular-expression syntax. For example, if $pat1 in the example above contains `:-)`, a fatal error will occur at runtime because `/:-/` is not a legal regular expression. (The quotemeta operator can be helpful in these cases – see the perLN func man page.) Second, when a match operator contains variables, the regular expression is recompiled each time that the match operator is used, generally resulting in slower performance. The `/o` (“compile once”) option causes a regular expression containing variables to be compiled only once:

```perl
($pat1, $pat2) =qw( (?i)\bjoseph\b (?i)\bhall\b); while (<FILEHANDLE>) {
    print if /$pat1/o or /$pat2/o;
}
```

To get this to work with arbitrary lists of patterns, though, you need to resort to some trickery. The usual method is to use a string `eval` returning an anonymous subroutine in combination with a `/o` match operator. This makes it possible to construct a list of anonymous subroutines, each of which searches its argument for a particular pattern:

```perl
@pats =qw( (?i)\bjoseph\b (?i)\bnathan\b (?i)\bhall\b); @search = map { eval q{ my $pat = $_; sub {$_[0] =~ /$pat/o } } @pats;
    while (defined($line = <FILEHANDLE>)) {
    for (@search) {
        if ($_->($line)) { $count++; last }
    }
    }
}
```

print "matches = $count
";

You could also construct a single pattern that matches an alternation of the original list of patterns. That might appear to be more efficient at first, but in my benchmarks it doesn’t seem to make a large difference.

If you are using Perl 5.005, an alternative (and more readily comprehensible) means of interpolating regular expressions is available through the `qr` (quote regex) operator. When 5.005 is widely adopted, `qr` will become the most appropriate mechanism for engineering solutions to this type of problem.

Reading Data into Nested Structures

You can handle some common tasks by reading data into one or two ordinary hashes, but for more complex analysis tasks you may need to use nested hashes and/or arrays. In order to work with nested data structures, you will need an understanding of reference syntax (too complicated to cover here, sorry!). You should also understand auto-vivification in Perl. Auto-vivification is a mechanism by which structures linked by ref-
ferences are created automatically. To illustrate, let’s suppose that the variable \$stats has the value \$undef. Now, consider the following line of Perl:

```perl
$stats->{\$host} = (Bytes => $bytes);
```

We are using \$stats like a hash reference. Even though \$stats is undefined, when we assign a value to \$stats->{\$host}, Perl will automatically create the underlying hash and assign a reference to it to \$stats. Now, \$stats->{\$host}{Bytes} will return whatever the value of \$bytes was. Auto-vivification also works for arbitrarily deeply nested structures. We could have written the above as:

```perl
$stats->{\$host}{Bytes} = $bytes;
```

And, in fact, that’s the idiomatic way to do it in Perl. Nested structures are useful when you must summarize or reorganize the data in a file. As an example, let’s look at analyzing httpd logs in Common Log Format (CLF). Let’s create a list of all the different hosts that connected to the Web server on each day, and print total bytes for each:

```perl
my \$log = 'access_log';
open LOG, \$log or die "Couldn’t open \$log: \$!";
my \%bytes;
while (<LOG>) {
  # split line into various fields
  my \$line;
  my (\$host, \$date, \$request, \$status, \$bytes) = \$line =~ /
    \([^:]*[/:]*[\.*]?([\:\.]*\[^:]*[/:]*\[^\.*][/:]*\[^\.*][/:]*\[^\.*]$/
    \?\[^\.*][/:]*\[^\.*]$/;) \n    \?\[^\.*][/:]*\[^\.*]$/; \n  # truncate host name to domain.domain if necessary
  \$host = \$host =~ /
    \([^\.*][/:]*\[^\.*]$/; \n    \?\[^\.*][/:]*\[^\.*]$/; \n  \$bytes =~ /
    \[^\.*][/:]*\[^\.*]$/; \n  \$bytes{\$date}{\$host} += \$bytes;
}
for my \$date (sort keys \%bytes) { 
  print \$date:
  for my \$host (sort \%bytes{\$date}keys) { 
    print "\$host: \$bytes{\$date}{\$host} bytes\n";
  }
}
```

The first part of this program (the while loop) reads in the log file a line at a time, extracting the various interesting parts of each line. (We aren’t using \$status or \$request here, but I left them in for clarity.) The hostname is cleaned up, and lines where no bytes were transferred are ignored; then the number of bytes is added to an “accumulator” in a nested hash. A transfer of 5,000 bytes on 02/Jan/1999 from a host named foo.bar would be added like this:

```perl
$bytes{'02/Jan/1999'}{'foo.bar'} += 5000;
```

Auto-vivification will create the appropriate underlying hashes and references anew if there is no existing entry for that date and/or host. The second part of the program sorts and prints out the dates and hostnames in a useful format, ordered first by date (alphabetically, for simplicity’s sake) and then in descending order by number of bytes transferred.

I’ll finish with one more example. This time, let’s look through the log and print out stats for the five largest transfers:

Auto-vivification also works for arbitrarily deeply nested structures. . . . Nested structures are useful when you must summarize or reorganize the data in a file.
my $log = '/etc/httpd/logs/access_log';
open LOG, $log or die 'Couldn't open $log: $!';
# initialize so -w is happy
my @largest = map { +( Bytes => 0 ) } 1..5;
while (<LOG>) {
    # split line into various fields
    my ($host, $time, $request, $status, $bytes) = /\S+\.*\S\.*\S\.*\S\.*\S\.*\S*/;
    # truncate host name to domain.domain if necessary
    ($host) = ($host =~ /\^[\.,\n]+?\.\n\^[\.,\n]+?\$/);
    next if $bytes =~ /\D/;  # skip if $bytes non-num, e.g. '-'
    # keep track of largest so far; re-sort if changed
    if ($largest[0]{Bytes} <= $bytes) {
        @largest = sort { $b->{Bytes} <=> $a->{Bytes} }
            @largest[0..3],
            { Host => $host, Time => $time, Request => $request,
                Bytes => $bytes
            }
    }
    for (@largest) {
        print "$_->{Host}: $_->{Bytes} bytes on $_->{Time}",
            " for request $_->{Request}\n";
    }
}

In this program we're using nested structures to keep track of information about a list of the largest transfers found so far. $largest[0] is a reference to a hash containing information (host, time, request, bytes) about the largest transfer seen so far, $largest[1] contains information about the second-largest one seen so far, and so on. Whenever a new, larger transfer is encountered, the new transfer is added to the list and the list is resorted.

Both of these programs run reasonably quickly – under a minute on 20MB log files on an older Sparc 20.

**Summary**

Perl is a powerful tool for analyzing and summarizing log files and other types of text databases. I've tried to show a few simple examples as well as some meatier ones. Of course, you don't always have to construct your own analysis code from scratch. There are CPAN modules that will help you analyze Web and other logs, so if you have a more complex analysis task, be sure to check there to see whether your problem might already be partially or completely solved for you.
the dark side of regular expressions

Recently, I was asked to look into a problem with a ksh script; it seemed to be hanging. The script looked innocuous, so I did the normal tracing thing via ksh -x. It seem to hang in a pipeline involving a sed command that was essentially doing a basename of the fourth field. The input file was all one line; the fields are separated by blanks. It looked like this:

```
$ systemx=: sed 1q $input
satla cserver
/gecko/rcv/compressed/IBMZZZZZ.YYYYYY.IBL94.AOHOLD.G1589V00.2030.19981119122414.mpde.mc.gz
/gecko/rcv/cn/IBMZZZZZ.YYYYYY.IBL94.AOHOLD.G1589V00.2030.19981119122414.mpde.mc.gz
mpde.mc 149266837 1181 Nov 19 12:24
```

So first I verified that the command was not hanging, but just taking a very long time. This also could give a baseline to evaluate any improvements we might make.

```
$ systemx=: sed 1000q $input | ptime sed 's:.*\([^ ]*\).*:\1:' > /dev/null
real 21:17.423
user 21:13.372
sys 0.071
```

This is grizzly. (The file was much larger than 1000 lines!) To reassure myself that it was a regular-expression problem, and not data related, I used awk to filter out the fourth field.

```
$ systemx=: sed 1000q $input | ptime awk '{print $4}' > /dev/null
real 0.079
user 0.061
sys 0.014
```

As we suspected, it wasn't data related. Let's try sed on the filtered text:

```
$ systemx=: sed 1000q $input | awk '{print $4}' | ptime sed 's:.*\([^ ]*\).*:\1:' > /dev/null
real 37.815
user 36.842
sys 0.057
```

Hmmm, a 4x improvement for processing 2.7x less data. This smells nonlinear. We can take advantage of the filtering to use a simpler pattern:

```
$ systemx=: sed 1000q $input | awk '{print $4}' | ptime sed 's:.*: : ' > /dev/null
real 0.165
user 0.147
sys 0.014
```

Yup, it looks like the backreferencing was the culprit all along. In general, backreferencing can take exponential time, but we rarely see such behavior. This time I guess we were lucky. Of course, now the pattern is so simple that we may as well do it all in awk:

```
$ systemx=: sed 1000q $input | ptime awk '(sub(\'.*/', '', $4); print $4)' > /dev/null
real 0.099
user 0.077
sys 0.014
```

Overall, the CPU (user) time is about 1700x faster, and as they say in the performance business, you eventually will notice factors of 1700. Ongoing, this change saved about six hours of CPU time per day. A good return on 10 minutes of real thought.
By now, most folks have heard about Tcl and Tk, but it seems only fair to introduce a new column with an introduction to the topic.

The Tcl package was developed by Dr. John Ousterhout in the late 1980s while he was at the University of California at Berkeley. He and his group were developing circuit simulators and found that each project needed a macro language to tune the system. After designing several on-the-fly macro languages, Dr. Ousterhout designed a package that could be merged into the other projects to provide a uniform language across the projects.

Since then, desktop computers have grown from “fast” 33 Mhz 386 processors to “slow” 300 Mhz Pentiums, and Tcl has grown from a simple embeddable macro language into a multipurpose package.

Describing the modern Tcl is a lot like describing the proverbial elephant.

If you look at Tcl from one angle, it’s a scripting language, like Perl, awk, or sh. If you look at the other side of Tcl/Tk, it’s a GUI programming language similar to Visual Basic, or a multiplatform language like Java. From another angle, Tcl is an interpreter that you can extend with your own commands (or existing libraries). Finally, Tcl is a language toolkit that you can merge into your program.

Just to add a bit to the indescribable nature of Tcl, it’s commercially supported freeware. The interpreters (including source code) are supported and made available for free from Scriptics (http://www.scriptics.com).

The core Tcl distribution comes with two interpreters: tclsh, a text-based interpreter suitable for text-oriented programs like CGI, shell scripts, or even client-server programs; and wish, the same base interpreter as tclsh extended with GUI graphics-oriented commands. There are also two libraries, one for Tcl and one for Tk, which let you build your own interpreter or merge Tcl/Tk into your application.

The ability to merge new object-code libraries into the interpreter is the feature that distinguishes Tcl from scripting languages like sh and awk. This feature lets you merge a vendor-supplied library (*.dll or *.a) right into the Tcl or Tk interpreter to create an interpreter with new task-specific commands.

Internet pioneer Einar Stefferud sometimes explains that the hallmark of a good Internet protocol is that it is simple at the core, with complexities at the edges. The most popular protocols (SMTP, HTTP, NNTP) follow a simple query/response format, with the complexity living in the message content, not the protocol.

Tcl follows a similar pattern: The core Tcl scripting language has a simple and regular syntax with a fairly small number of commands. The complex edges, in this case, are the extensions. The interpreter extensions have new commands that interact with a new object code library, while the language core stays the same.

Using Tcl makes it easy to move from problem domain to problem domain. You don’t need to learn a whole new language. You just need to learn the new commands for that application.

Several years ago I proved that a novice (me) armed with John Ousterhout’s book could create an interpreter with a new set of commands in one evening. Now, with the discussions of extension building in books from Brent Welch, J. A. Zimmer, and myself, the learning curve may be shorter.
The Tcl syntax is trivial:

- The first word on a line is a command name.
- Words are separated by whitespace.
- Words can be grouped with curly braces ({}), or quotes (" ").
- A line can be continued across several lines by escaping the newline with a backslash.
- Any words in a command line after the first word are arguments to the command.
- A variable name preceded by a dollar sign ($) is replaced by the value of the variable.
- A command within square brackets is replaced by the results of evaluating that command. This is similar to how backquotes (') are handled by shell scripts.
- A comment starts with a "#" symbol.

So, let's take a cursory look at some Tcl commands. With just seven commands, we can build a GUI-based calculator.

**Assigning Values to Variables**

Probably the most used command in Tcl is set. The set command assigns a value to a variable.

**Syntax:**

```
set foo "bar";     # Assigns the string "bar" to variable foo.
set pi 3.1415;    # Assigns the value 3.1415 to the variable pi.
set x a b;       # An illegal operation, only one value can be set.
```

Tcl also enables you to append a new string to the value already in a variable. This is done with the append command.

**Syntax:**

```
append foo "baz"; # Appends the string "baz" to variable foo.
append pi 19;     # Add two digits of accuracy to the previous value of pi.
```

**Performing Math**

The command to perform arithmetic operations is expr, which behaves like the Bourne shell expr command. Tcl supports all the math calls in the standard C math library, including the trig and exponential functions.

**Syntax:**

```
set twoPi [expr $pi * 2];           # set the variable 2 pi.
set circumference [expr $twoPi * $radius]; # circumference is 2*pi*radius
set area [expr pow($radius, 2) * $pi]; # area is pi * radius ^ 2
```

One extremely common math operation is simply incrementing or decrementing a variable by an integer. To make life a bit simpler, Tcl has a special command for adding a value to a variable: incr.

**Syntax:**

```
incr variableName value
set x 2    # Set x to 2
incr x 4   # Add 4 to the value of x. X is now 6
incr x -2  # Subtract 2 from the value of x. X is now 4.
```
Tk supports three layout managers to specify how your application should look on the screen.

Looping

Tcl supports a loop-on-counter construct (for), a loop-on-test construct (while), and a loop-on-list-contents construct (foreach).

The calculator example uses only foreach, so that’s all I’ll describe here.

Syntax: foreach variableName list { body }

The foreach command will iterate through the values in the list. It will evaluate the body after setting the value of the loop variable to the appropriate list element for this iteration.

    # Initialize the total to 0
    set total 0;
    # For each value in the list '2 4 9'
    # add that value to the previous value of total
    foreach value { 2 4 9 } {
        incr total $value
    }

GUI Widgets

Tk supports many graphic widgets for building GUIs, including a drawable canvas, an editable text window, and a picture object that supports simple image operations. For this example, we’ll just need two widgets and a geometry manager.

The widget creation commands all follow a common format:

Syntax: widgetType widgetName ?arguments?

The widgetType is the type of widget to create: button, label, canvas, etc.

The widgetName is a name for this specific instance of the widget. The naming convention for Tk widgets is that widget names must be unique and must start with a period/lowercase letter pair.

The arguments enable you to specify widget configuration options like the text to display, the foreground and background color, and size of margins. These are defined as -optionName value pairs.

All parameters of a Tk widget can be set when the widget is created and also modified after a widget exists. However, unlike programming with the X library, the widgets have a set of good defaults, so you don’t need to define all the parameters when you create a widget.

Button

One common GUI widget is the button widget. This widget will display a string (or graphic) and perform an action when the button is clicked.

Syntax: button .buttonName ?arguments?

Two commonly used arguments are:

-text string The text to display on the button.
-
-command body The body of a command to evaluate when the button is activated.

Label

A label simply displays a string. One of the neat features of the Tk label is that you can link the label to a variable, and it will automatically display the contents of that variable. Your code doesn’t need to do anything to update the display.
Syntax: label labelName ?arguments?

-textvariable variable Name This label will display the contents of the named
variable.

-text string This label will display a particular string.

Grid
Tk supports three layout managers to specify how your application should look on the
screen. For the calculator example, the "grid" manager, which lays out widgets in a
spreadsheet style, is the simplest to use. The grid command defines where a widget will
appear and maps the widget onto the display.

Syntax: grid widgetName ?arguments?

-row rowNumber The row for this widget.
-col columnNumber The column for this widget.
-columnspan number The number of columns this widget will use.

A Calculator
With these seven commands, we can construct a little GUI calculator. Now, this is not
the last word in online calculators, but it's an example of how little code you need to
create a useful Tcl/Tk application. Complete with comments, this is 50 lines of code.

```
# Initialize a string that will contain the math operations
# to perform
set math ""
# Initialize a position counter for the widgets being created.
set pos 0
# Loop through the numbers and operations creating buttons for
# each widget
foreach val {1 2 3 4 5 6 7 8 9 + - / } {
    # Create a button
    # The button names are .b0, .b1, .b2, etc.
    # The text to display is the current item in the list
    # The action for the button is to append that value onto the
    # math string
    button .b$pos -text $val -command 'append math $val'
    # The buttons are displayed in a 3-column-wide grid
    # Calculate the row and column
    set row [expr ($pos) / 3]
    set col [expr ($pos) % 3]
    # And map the widget to the screen
    grid .b$pos -row $row -col $col
    # Increment the position/name counter
    incr pos
}
# The equals button has a different command.
# When the equals button is clicked, it will evaluate the math
# expression, and assign the output to the result variable.
# It then clears the math expression for the next set of
# calculations.
button .b_eq -text '=' -command {set result [expr $math];
    set math '"
}
grid .b_eq -row 4 -col 2
# Create two labels to display the math expression and result.
label .math -textvariable math
grid .math -row 5 -column 0 -columnspan 3
label .result -textvariable result
grid .result -row 6 -column 0 -columnspan 3
```
Learning More

If you don't already know Tcl/Tk, you are (I hope) interested in learning a bit more by now. Here are a few books and Web sites that will get you started.

The definitive book, but somewhat dated.

An excellent book for the experienced programmer.

A good introductory book.

I think it's a good book, but I may be biased.

Here are some sites with general Tcl/Tk information:

<http://www.scriptics.com>
The Scriptics home page. Up-to-date information on the state of Tcl; free source code; supported binary downloads; for-sale development utilities; training, support, and pointers to Tcl/Tk resources.

<http://www.tclconsortium.org>
The Tcl/Tk Consortium home page. The Tcl/Tk Consortium is a nonprofit organization of Tcl advocates with a charter to make Tcl known and available to the computing community. The Web site includes links to resources; information; and a chance to buy precompiled versions of Tcl and Tcl extensions for popular platforms.

One of the best collections of pointers to Tcl "stuff," ranging from discussions of Tcl fine points to tutorials, books, articles and FAQs.

And, finally, some sites with online or CAI Tcl/Tk instruction:

<http://www.mssn.com/~clif/TclTutor.html>
The TclTutor interactive computer-based training package for Win 95/NT, UNIX, and Macintosh.

<http://hegel.itc.ukans.edu/topics/tcltk/tutorial-noplugin/index.html>
Robert Hill, Shyamalan Pather, and Matt Peters created this 13-lesson tutorial on the Tcl language.

<http://www.dci.cica.ac.uk/Publications/Cookbook/index.html>
This is an excellent and complete tutorial by Lakshmi and Venkat Sastry. It covers Tcl, Tk, and building extensions.

<http://www.cujo.com/tcl_tut.html>
William Ho (<bill@technologyarchitects.com>) has written a concise introduction to the Tcl language.
electronic snake oil

So your company has finally decided to heed the advice you gave them 11 years ago, and they're going to start using email to communicate with business partners. Or, more likely, they've already been doing that, but now they have concerns about the security of doing so. Specifically, they don't want competitors reading the memos that are going back and forth. Or maybe you're sending email to your aunt Lena who lives in Russia and you don't think what you have to tell her is the business of anyone at Ft. Meade. Or your company has a number of small offices around the country that you want to tie together through a virtual private network (VPN) over the Internet.

In any of these cases, you have a generic need to send data to another party privately. You need to lock your data. You need cryptography. But how does a system administrator, or some other technical person who isn't a cryptographer, know the difference between a good cryptography product and the stuff of Cracker Jack secret decoder rings? The short answer is that there isn't an any really easy answer. However, over the years, a few common practices have developed that have helped us identify the traits of those whose products are "snake oil." Before we consider those, though, let's cover some terminology and basic concepts of cryptography.

Snake Oil and Silver Bullets

Why "snake oil"? In many fields, the term is used to denote something sold without consideration of its quality or its ability to fulfill its vendor's claims. This term originally applied to elixirs sold in traveling medicine shows. The salesmen would claim their elixir would cure just about any ailment that a potential customer could have. Listening to the claims made by some crypto vendors, "snake oil" is a surprisingly apt name.

Basic Concepts

A wide variety of information on cryptography is available. There is the USENET Cryptography FAQ, RSA's Cryptography Today FAQ, and books such as Bruce Schneier's excellent *Applied Cryptography* [1].

When evaluating any product, be sure to understand your needs. For data-security products, what are you trying to protect? Do you want a data archiver, an email plugin, or something that encrypts online communications? Do you need to encrypt an entire disk or just a few files? And how secure is secure enough? Does the data need to be unreadable by "spies" for five minutes, one year, or 100 years? Is the spy someone's kid sister, a corporation, or a government?

Symmetric versus Asymmetric Cryptography

There are two basic types of cryptosystems: symmetric (also known as "conventional" or "secret key") and asymmetric ("public key").

Symmetric ciphers require both the sender and the recipient to have the same key. This key is used by the sender to encrypt the data and again by the recipient to decrypt the data. The problem here is getting the sender and recipient to share the key.

Asymmetric ciphers are much more flexible from a key-management perspective. Each user has a pair of keys: a public key and a private key. Messages encrypted with one key can only be decrypted by the other key. The public key can be published widely, while...
the private key is kept secret. So if Alice wishes to send Bob some secrets, she simply finds and verifies Bob's public key, encrypts her message with it, and mails it off to Bob. When Bob gets the message, he uses his private key to decrypt it. Verification of public keys is an important step. Failure to verify that the public key really does belong to Bob leaves open the possibility that Alice is using a key whose associated private key is in the hands of an enemy.

Asymmetric ciphers are much slower than their symmetric counterparts. Also, key sizes generally must be much larger.

**Secrecy versus Integrity: What Are You Trying to Protect?**

For many users of computer-based crypto, preserving the contents of a message is as important as protecting its secrecy. Damage caused by tampering can often be worse than damage caused by disclosure. For example, it may be disquieting to discover that a cracker has read the contents of your funds-transfer authorization, but it's a disaster for him to change the transfer destination to his own account.

Encryption by itself does not protect a message from tampering. In fact, there are several techniques for changing the contents of an encrypted message without ever figuring out the encryption key. If the integrity of your messages is important, don't rely on secrecy alone to protect them. Check how the vendor protects messages from undetected modification.

**Key Sizes**

Even if a cipher is secure against analytical attacks, it will be vulnerable to brute-force attacks if the key is too small. In a brute-force attack, the attacker simply tries every possible key until the right one is found. How long this takes depends on the size of the key, how computationally intensive the encryption (or decryption) process is, and the amount of processing power available. So when trying to secure data, you need to consider how long it must remain secure and how much computing power an attacker can use.

Some guidelines have been offered for choosing an appropriate key length. For instance, Table 1 shows the cost of breaking symmetric keys by brute force, as noted by Blaze et al.[2]. This report strongly recommends using symmetric keys of 90 bits or more.

With the tremendous increases in computing power over the last several decades, cryptosystems once considered secure are now vulnerable to brute-force attacks. RSA Laboratories sponsored a series of contests, collectively known as the 1997 Secret Key Challenge[5]. So far, we have seen RC5 up to 56 bits fall victim to brute-force attacks, as well as the financial industry's workhorse, DES. At 56 bits, the keys used for DES are just too small to stand up to a dedicated attacker. It's noteworthy that both of the first two groups to break a DES-encrypted message did so with essentially no funding. The Electronic Frontier Foundation funded a third break, performed with a special-purpose DES-cracking machine, "Deep Crack," which did the job in 56 hours.

If small nonprofit groups can fund development of a machine like Deep Crack, certainly the power that exists in larger for-profit organizations, organized crime, and government intelligence agencies can go well beyond 56 bits.
Table 1: Time and Cost of Key Recovery

<table>
<thead>
<tr>
<th>Type of Attacker</th>
<th>Budget</th>
<th>Tool</th>
<th>Time and Cost per 40-bit Key Recovered</th>
<th>Key-length Needed for Protection in Late 1995</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pedestrian hacker</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tiny</td>
<td>$400</td>
<td>Scavenged Computer Time</td>
<td>1 Week ($0.08)</td>
<td>45</td>
</tr>
<tr>
<td>$10,000</td>
<td>FPGA</td>
<td>5 Hours ($0.08)</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td><strong>Small business</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10,000</td>
<td>FPGA</td>
<td>12 Minutes ($0.08)</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td><strong>Corporate Department</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$300K</td>
<td>FPGA</td>
<td>24 seconds ($0.08)</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td><strong>Big Company</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10M</td>
<td>ASIC</td>
<td>.005 seconds ($0.001)</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td><strong>Intelligence Agency</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$300M</td>
<td>ASIC</td>
<td>.0002 seconds ($0.001)</td>
<td>75</td>
<td></td>
</tr>
</tbody>
</table>

As mentioned earlier, asymmetric ciphers typically require significantly longer keys to provide the same level of security as symmetric ciphers. Comparing key lengths between algorithms is awkward because different algorithms have different characteristics. Knowing the key size is useless if you don’t know what type of algorithm is being used.

But to give you some idea of what’s reasonable, Table 2 [1] compares symmetric keys against one type of asymmetric key: those based on the “factoring problem” or the “discrete log problem.” (Algorithms based on the “elliptical curve discrete log problem” are more resistant to brute-force attacks and can use much smaller keys. In fact, they don’t have to be much larger than symmetric keys, as far as is known now.)

### Keys versus Passphrases

A “key” is not the same thing as a “passphrase” or “password.” In order to resist attack, all possible keys must be equally probable. If some keys are more likely to be used than others, then an attacker can use this information to reduce the work needed to break the cipher.

Essentially, a key must be random. However, a passphrase generally needs to be easy to remember, so it has significantly less randomness than its length suggests. For example, a 20-letter English phrase, rather than having 20 x 8 = 160 bits of randomness, only has about 20 x 2 = 40 bits of randomness. So, most cryptographic software will convert a passphrase into a key through a process called “hashing” or “key initialization.” Avoid cryptosystems that skip this phase by using a password directly as a key. Avoid anything

Table 2: Key Lengths with Similar Resistance to Brute-Force Attacks

<table>
<thead>
<tr>
<th>Symmetric Key Length</th>
<th>Public Key Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>56 bits</td>
<td>384 bits</td>
</tr>
<tr>
<td>64 bits</td>
<td>512 bits</td>
</tr>
<tr>
<td>80 bit</td>
<td>768 bits</td>
</tr>
<tr>
<td>112 bits</td>
<td>1792 bits</td>
</tr>
<tr>
<td>128 bits</td>
<td>2304 bits</td>
</tr>
</tbody>
</table>
Perhaps the biggest warning sign of all is the “trust us, we know what we’re doing” message that's either stated directly or implied by a vendor. That doesn't let you generate your own keys (e.g., the vendor sends you keys in the mail, or keys are embedded in the copy of the software you buy).

**Implementation Environment**

Other factors that can influence the relative security of a product are related to its environment. For example, in software-based encryption packages, is there any plaintext that's written to disk (perhaps in temporary files)? What about operating systems that have the ability to swap processes out of memory onto disk? When something to be encrypted has its plaintext counterpart deleted, is the extent of its deletion a standard removal of its name from the directory contents, or has it been written over? If it's been written over, how well has it been written over? Is that level of security an issue for you? Are you storing cryptographic keys on a multi-user machine? If so, the likelihood of having your keys illicitly accessed is much higher. It's important to consider such things when trying to decide how secure something you implement is (or isn't) going to be.

**Snake Oil Warning Signs**

**“Trust Us, We Know What We’re Doing”**

Perhaps the biggest warning sign of all is the “trust us, we know what we're doing” message that's either stated directly or implied by a vendor. If the vendor is concerned about the security of their system after describing exactly how it works, it is certainly worthless. Regardless of whether or not they tell, smart people will be able to figure it out. The bad guys know your secrets (especially if you are an especially attractive target, such as a large company, bank, etc.) are not stupid. They will figure out the flaws. If the vendor won't tell you exactly and clearly what's going on inside, you can be sure that they're hiding something, and that the only one to suffer as a result will be you, the customer.

**Technobabble**

If the vendor’s description appears to be confusing nonsense, it might very well be so, even to an expert in the field. One sign of technobabble is a description that uses newly invented terms or trademarked terms without actually explaining how the system works. Technobabble is a good way to confuse a potential user and to mask the vendor’s own lack of expertise.

And consider this: If the marketing material isn’t clear, why expect the instruction manual to be any better? Even the best product can be useless if it isn’t applied properly. If you can’t understand what a vendor is saying, you’re probably better off finding something that makes more sense.

**Secret Algorithms**

Avoid software that uses secret algorithms. This is not a safe means of protecting data. If the vendor isn’t confident that its encryption method can withstand scrutiny, then you should be wary of trusting it.

A common excuse for not disclosing an algorithm is that “hackers might try to crack the program's security.” While this may be a valid concern, it should be noted that such “hackers” can reverse-engineer the program to see how it works anyway. This is not a problem if the algorithm is strong and the program is implemented properly.

Using a well-known trusted algorithm, providing technical notes explaining the implementation, and making the source code available are signs that a vendor is confident.
about its product’s security. You can take the implementation apart and test it yourself. Even if the algorithm is good, a poor implementation will render a cryptography product completely useless. However, a lock that attackers can’t break even when they can see its internal mechanisms is a strong lock indeed. Good cryptography is exactly this kind of lock.

Note that a vendor who specializes in cryptography may have a proprietary algorithm that it will reveal only under a nondisclosure agreement. The crypto product may be perfectly adequate if the vendor is reputable. (But how does a nonexpert know if a vendor is reputable in cryptography?) In general, you’re best off avoiding secret algorithms.

Revolutionary Breakthroughs

Beware of any vendor who claims to have invented a “new type of cryptography” or a "revolutionary breakthrough." True breakthroughs are likely to show up in research literature, and professionals in the field typically won’t trust them until after years of analysis, when they’re not so new anymore.

The strength of any encryption scheme is only proven by the test of time. New crypto is like new pharmaceuticals, not new cars. And in some ways it’s worse: If a pharmaceutical company produces bogus drugs, people will start getting sick, but if you’re using bogus crypto, you probably won’t have any indication that your secrets aren’t as secret as you think.

Avoid software that claims to use “new paradigms” of computing such as cellular automata, neural nets, genetic algorithms, chaos theory, etc. Just because software uses a different method of computation, it isn’t necessarily more secure. (In fact, these techniques are the subject of ongoing cryptographic research, and nobody has published successful results based on their use yet.)

Also be careful of specially modified versions of well-known algorithms. This may intentionally or unintentionally weaken the cipher.

It’s important to understand the difference between a new cipher and a new product. Engaging in the practice of developing both ciphers and cryptographic products is a fine thing to do. However, to do both at the same time is foolish. Many snake-oil vendors brag about how they do this, despite the lack of wisdom in such activity.

Experienced Security Experts, Rave Reviews, and Other Useless Certificates

Beware of any product that claims it was analyzed by “experienced security experts” without providing references. Always look for the bibliography. Any cipher that they’re using should appear in a number of scholarly references. If not, it’s obviously not been tested well enough to prove or disprove its security.

Don’t rely on reviews from newspapers, magazines, or television shows, since they generally don’t have cryptographers to analyze software for them. (Celebrity “hackers” who know telephone systems are not necessarily crypto experts.)

The fact that a vendor is a well-known company or the algorithm is patented doesn’t make it secure either.

Unbreakability

Some vendors will claim their software is “unbreakable.” This is marketing hype and a common sign of snake oil. No algorithm is unbreakable. Even the best algorithms are susceptible to brute-force attacks, though this can be impractical if the key is large enough.
Some companies that claim unbreakability actually have serious reasons for saying so. Unfortunately, these reasons generally depend on some narrow definition of what it means to "break" security. For example, one-time pads (see the next section) are technically unbreakable as far as secrecy goes, but only if several difficult and important conditions are true. Even then, they are trivially vulnerable to known plaintext attacks on the message's integrity. Other systems may be unbreakable only if one of the communicating devices (such as a laptop) isn't stolen. So be sure to find out exactly what the "unbreakable" properties of the system are, and see if the more breakable parts of the system also provide adequate security.

Often, less-experienced vendor representatives will roll their eyes and say, "Of course it's not unbreakable if you do such-and-such." The point is that the exact nature of "such and such" will vary from one product to another. Pick the one that best matches your operational needs without sacrificing your security requirements.

**One-Time Pads**

A vendor might claim the system uses a one-time-pad (OTP), which is provably unbreakable. Technically, the encrypted output of an OTP system is equally likely to decrypt to any same-size plaintext. For example,

```
598v *$+_--- xCtMB0
```

has an equal chance of decrypting to any of these:

- the answer is yes
- the answer is no!
- you are a weenie!

Snake-oil vendors will try to capitalize on the known strength of an OTP. But it is important to understand that any variation in the implementation means that it is not an OTP and has nowhere near the security of an OTP.

An OTP system works by having a "pad" of random bits in the possession of both the sender and recipient, but absolutely no one else. Originally, paper pads were used before general-purpose computers came into being. The pad must be sent from one party to the other securely, such as in a locked briefcase handcuffed to the carrier.

To encrypt an n-bit message, the next n bits in the pad are used as a key. After the bits are used from the pad, they're destroyed and can never be used again. The bits in the pad cannot be generated by an algorithm or cipher. They must be truly random, using a real random source such as specialized hardware or radioactive decay timings. Some snake-oil vendors will try to dance around this issue and talk about functions they perform on the bit stream, things they do with the bit stream versus the plaintext, or something similar. But this still doesn't change the fact that anything that doesn't use real random bits is not an OTP. The important part of an OTP is the source of the bits, not what one does with them.

OTPs are seriously vulnerable if you ever reuse a pad. For instance, the NSA's VENONA project[4], without the benefit of computer assistance, managed to decrypt a series of KGB messages encrypted with faulty pads. It doesn't take much work to crack a reused pad.

The real limitation to practical use of OTPs is the generation and distribution of truly random keys. You have to distribute at least one bit of key for every bit of data trans-
mitted. So OTPs are awkward for general-purpose cryptography. They're only practical for extremely low-bandwidth communication channels where two parties can exchange pads with a method different from what they use to exchange messages. (It is rumored that a link from Washington, D.C., to Moscow was encrypted with an OTP.)

Further, if pads are provided by a vendor, you cannot verify the quality of the pads. How do you know the vendor isn't sending the same bits to everyone? Keeping a copy for themselves? Or selling a copy to your rivals? Also, some vendors may try to confuse random-session keys or initialization vectors with OTPs.

**Algorithm or Product X is Insecure**

Be wary of anything that claims that competing algorithms or products are insecure, without providing evidence for these claims. Sometimes attacks are theoretical or impractical, requiring special circumstances or massive computing power over many years, and it's easy to confuse a layman by mentioning these.

**Recovable Keys**

If there is a key-backup or key-escrow system, are you in control of the backup or does someone else hold a copy of the key? Can a third party recover your key without much trouble? Remember, you have no security against someone who has your key.

If the vendor claims it can recover lost keys without using some type of key-escrow service, avoid it. The security is obviously flawed.

**Exportable from the US**

If the software is made in the US, can it be exported? Strong cryptography is considered dangerous munitions by the United States and requires approval from the US Bureau of Export Administration, under the US Department of Commerce, before it can leave the country. Various interested government agencies serve as consultants to the Bureau of Export Administration when evaluating such requests. (The US isn't alone in this; some other nations have similar export restrictions on strong cryptography.) Chances are, if the software has been approved for export, the algorithm is weak or crackable.

If a vendor is unaware of export restrictions, avoid its software. For example, if it claims that the IDEA cipher can be exported, when most vendors (and the US Government!) do not make such a claim, then the vendor is probably lacking sufficient clue to provide you with good cryptography.

Because of export restrictions, some decent crypto products come in two flavors: US-only and exportable. The exportable version will be crippled, probably by using smaller keys, making it easy to crack.

There are no restrictions on importing crypto products into the US, so a non-US vendor can legally offer a single, secure version of a product for the entire world.

Note that a cryptosystem may not be exportable from the US even if it is available outside the US. Sometimes a utility is illegally exported and posted on an overseas site.

**Military Grade**

Many crypto vendors claim their system is "military grade." This is a meaningless term, since there isn’t a standard that defines "military grade," other than being in use by armed forces. Since these organizations don’t reveal what crypto they use, it isn’t possible to prove or disprove that something is "military grade."
Unfortunately, some good crypto products also use this term. Watch for this in combination with other snake-oil indicators, such as “our military-grade encryption system is exportable from the US!”

Other Considerations

Avoid vendors who don’t seem to understand anything described in the “Basic Concepts” section above.

Avoid anything that allows someone with your copy of the software to access files, data, etc., without needing some sort of key or passphrase.

Beware of products that are designed for a specific task, such as data archiving, and have encryption as an additional feature. Typically, it’s better to use an encryption utility for encryption, rather than some tool designed for another purpose that adds encryption as an afterthought.

No product is secure if used improperly. You can be the weakest link in the chain if you use a product carelessly. Do not trust any product to be foolproof, and be wary of any product that claims it is.

Interface isn’t everything: user-friendliness is important, but be wary of anything that puts too much emphasis on ease of use without due consideration to cryptographic strength.

Glossary

algorithm A procedure or mathematical formula. Cryptographic algorithms convert plaintext to and from ciphertext.

cipher Synonym for “cryptographic algorithm”

escrow A third party able to decrypt messages sent from one person to another. Although this term is often used in connection with the US Government’s “Clipper” proposals, it isn’t limited to government-mandated ability to access encrypted information at will. Some corporations might wish to have their employees use cryptosystems with escrow features when conducting the company’s business, so the information can be retrieved should the employee be unable to unlock it himself later (if he were to forget his passphrase, suddenly quit, get run over by a bus, etc.) Or, someone might wish her spouse or lawyer to be able to recover encrypted data, etc., in which case she could use a cryptosystem with an escrow feature.

initialization vector One of the problems with encrypting such things as files in specific formats (i.e., that of a word processor, email, etc.) is that there is a high degree of predictability about the first bytes of the message. This could be used to break the encrypted message more easily than by brute force. In ciphers where one block of data is used to influence the ciphertext of the next (such as CBC), a random block of data is encrypted and used as the first block of the encrypted message, resulting in a less predictable ciphertext message. This random block is known as the initialization vector. The decryption process also performs the function of removing the first block, resulting in the original plaintext.

key A piece of data that, when fed to an algorithm along with ciphertext, will yield plaintext (or, when fed to an algorithm along with plaintext, will yield ciphertext).
a management perspective on privileged access to computer systems

Many computer systems define some kind of privileged access, to be sure that certain sensitive functions are protected and to allow system staff to override security and other protections in the course of doing their jobs. (UNIX has a “superuser” called root, which has total control of the system; VAX/VMS has the bypass bit; Windows NT has the Administrator user. Other systems have similar things, by various names.) Yet the very term “privileged access” presumes the existence of unprivileged access, the kind granted to most regular, authorized users, which some of them find demeaning and overly restrictive. The determination of who gets privileged access is usually made by the systems staff. It should be made fairly and reasonably, according to policies that apply to everyone. But sometimes conflicts arise, and management must be prepared to resolve them.

The Question

Users who need to do something on the computer requiring privileged access they don’t have often pose the question, “Can I have the root password?” The first request is

References


There are usually ways to accomplish whatever the user must do short of granting unlimited privileged access. . . . But sometimes the true reason is, “I need it to protect my position” or “I need it to enhance my self-esteem.”

seldom granted without deeper probing; the systems staff asks, “Why? What for?” and the user has one of several typical responses. The commonest, and easiest to handle, is “I need it to do A, B, and C.” There are usually ways to accomplish whatever the user must do short of granting unlimited privileged access, and the systems staff can direct the user to those tools or methods, up to and including doing something manually on the user’s behalf.

But sometimes the true reason is, “I need it to protect my position” or “I need it to enhance my self-esteem.” Getting to the truth of such reasons can be difficult and the source of much of the conflict that can arise. In these cases, it is vital that management get involved to resolve the issue without compromising the organization’s goals. Managers must understand the principles involved to make the right decision.

The Stability Argument

Computer systems are very complex, the more so as they become more flexible and powerful, and therefore more valuable to their owners. Even a single-user personal computer is complex enough to demand a great deal of its user’s time and effort. Consider the typical workplace computing environment, which is a network of dozens, sometimes thousands, of computers, all interacting in various ways. There’s a lot going on.

Software developers, when they create these systems, have many choices to make about how the systems will operate. But to reach the largest market, they leave many of those choices up to the customer, calling their systems “flexible” and “policy-neutral.” And indeed they are, but someone has to flex them, and someone has to set the policies. And someone has to tell the software what the policies are, to configure things so they interact correctly.

Such configuration is invariably central to the correct operation of the system as a whole. There is a high risk of a mistake having widespread effects. System configuration is typically protected, requiring privileged access to change things, so (the developers imagine) only those who know what they are doing will change anything. The system will run smoothly, according to the developers’ vision, and the users will be productive and happy.

The stability argument says that most users cannot know all the details of how the system is configured and implemented, so they cannot always make informed choices about what to change. A little knowledge is a dangerous thing. Even a lot of knowledge may not be enough to avoid making a serious mistake. And granting unlimited privileged access gives the user the ability, if not the motive, to make such changes.

Often, the mistakes happen far away from their effects. In one example, users were having trouble logging in “sometimes.” Their accounts seemed to go in and out of existence in the space of hours. Some days the problem got worse, some days better. After almost a week, the senior sysadmin tracked the problem to a transient machine set up as an NIS server, brought to the headquarters site by visitors from a field office in Israel. Clients find an NIS server by broadcasting on the network, and sometimes they found this bogus one instead of the authorized ones run by the local staff. The NIS domain name was the same, a fact that caused no trouble back in the field office because broadcasts did not reach across the wide-area network. The mistake had its roots months in the past and a continent away. Privileged access to that one machine, and its later transport, was all that was needed.
In another example, a remote office (this one in Paris) was configured to have an incorrect so-called default route, normally directing network packets toward the corporate headquarters and its Internet gateway. But this machine was also configured to run a dynamic routing process, which told all the other computers on the network this route, sucking the entire company’s Internet-bound packet traffic into a black hole for about 24 hours.

These are simple examples with fairly easy corrective actions (such as restricting where core routers get their default route). But more examples are easily found of “accidents” with profoundly obscure mechanisms and highly non-local effects. The more complex the system or network, the worse this gets, and the harder it is for even the specialists to avoid making disastrous mistakes.

In these scenarios, not having root privileges means not having a sword hanging over one’s head. Privilege carries responsibility: privilege one may not need, and responsibility one can probably do without. If there’s a way to avoid it, take that way.

**The Productivity Argument**

Computers are supposed to be tools. Some people collect and enjoy tools for their own sake, but a carpenter carries a hammer to pound in nails and build something, not just because it looks good on the belt.

There are two kinds of computer users: those who do computing for its own sake (system staff and programmers); and those who use the computers to accomplish something unrelated to the computers they use, something you are paying them to do. The more time they spend in overhead, getting the tools to work correctly, the less time they spend... not “doing work” exactly, but getting their work done.

With the complexity of the tools comes specialization. Even in an office where “people administer their own desktops” (a codephrase for disaster in this context), an informal leader will often emerge, the power user to whom the others turn when they have a problem. (Who helped you when had trouble upgrading your laptop? That’s the one.) After a while, that person will obtain the secret passwords to the file server or router or whatever, and, if you’re lucky, will keep them secret. You now have a sysadmin.

But what was that person’s “real” job, again? It’s getting worked on, to be sure, but is it getting done? Are you happy with that person’s performance in that respect?

If you’re less lucky, the informal leader will recognize this trap and avoid helping the others (and taking the fall at review time), and the rest of them will founder. Where’s the manual for A? How did you get B to print? C doesn’t work at all, so I (faxed it / wrote it to a floppy / bought a box of pencils / lost the order).

Specialization requires time and energy. Not everyone has, or should spend, the time to read all the manuals and figure out the systems and solve the system problems. But someone should.

One person who is knowledgeable about the systems and can set them up so they run well provides leverage for the productivity of all the other people there. Even in extremely small organizations, where a whole person can’t be justified, part of one should be officially recognized and assigned this role and the resources to carry it out. To let this function go unstaffed is to waste a great deal of the power in the systems, to waste much of the large investment in technology that most businesses now make. And, often much bigger, to waste expensive labor, misdirected into fooling with the computers while not getting the right value out of them.
How long would you stay in business if your employees could give themselves a raise by modifying the payroll database without authorization? Yet someone has to be able to modify it. That’s the crux of security.

In larger shops, where several people are dedicated to systems support, the problem is slightly different: There get to be too many specialists, too many cooks spoiling the broth. In the small shop, the support person will often have unofficial deputies who have privileged access and make limited changes, but communication is easy: “While you were on vacation, I rebooted D twice, and had to remove lock file E.”

In larger shops, it becomes ever more important to limit the absolute number of people with privileged access. Even informal deputies (former sysadmins, senior systems developers) become first unnecessary, and then undesirable; they have to be deputized formally. Change-control procedures come to the fore, and documentation is vital. In these situations, merely knowing about the procedures and how to follow them constitutes special knowledge that very few part-time helpers can be expected to maintain. And the folklore that somehow never gets documented underscores this. The answer to The Question becomes “Thank you for your offer of help, but we’ve got things under control.” It’s often not entirely true.

The Security Argument

Another seldom-admitted-to reason behind a request for more privileges is, “Because I feel I deserve to be trusted with this.” There is a fallacy behind this, to wit, that regular, authorized users aren’t trusted already. Virtually all information security systems depend absolutely on their users to cooperate with, if not enforce, the rules. All such systems can be thwarted by authorized users through malice, negligence, or just ignorance.

But of course there are degrees. If one has privileged access, one can bypass user identification. That is, one can masquerade not only as the “super” user, but as any other authorized user. One could then do bad things that would be ascribed to the victim. People have lost their jobs, even their freedom, from such acts. Those with this access are perforce trusted not to abuse it this way.

Security threats fall into three broad categories: access, disclosure, and modification. Unauthorized access is akin to breaking and entering (still a crime, even if the would-be burglar doesn’t take anything). Denial-of-service attacks are in this group, since they deny access to authorized users. With disclosure, files (say) don’t have to be destroyed to do the damage: Someone sends your email about that top-secret unannounced merger to the Wall Street Journal. Modification (especially if undetected) and its limiting case, destruction, are another group. How long would you stay in business if your employees could give themselves a raise by modifying the payroll database without authorization? Yet someone has to be able to modify it. That’s the crux of security.

Most computer systems feature some kind of user identification, at least as part of authorization. (What you can do depends on who you are.) Some don’t even do that; personal-computer operating systems are only now starting to get these features. Yet, with the concept of the all-powerful privileged user who can bypass everything, user identification isn’t perfectly reliable. It is meaningful only insofar as you control tightly who can bypass it.

“My people administer their own desktops, and everyone knows the root password to the fileserver. As long as we all know this, and agree that there is no security, what’s the problem?” The problem here is that you have discarded this feature of the operating system, and these hosts cannot be trusted with certain information as a result—information that people need to store and use in a trusted environment. Do you want to conduct all your secret communications about that merger only in person? What would
the airfare be like for those last-minute meetings? Okay, you need to trust the telephone. What about the fax machine? The network printer? The computer on your desk? Can you? Are you sure? What if you turn out to be uninformed about an important risk? What if you learn this from the Wall Street Journal?

The sysadmins aren’t supposed to be aware of the merger yet, but they can read your email (if they have the time and inclination to look, which is unlikely). But if there are only three of them, for sure, and the fact leaks from your office, the investigators will thank you for keeping the number small. Interviewing 40 people in this situation, six of whom aren’t even employees, is a lot harder, they will tell you. You can’t drive this number to zero, with the technology we’ve got today, but you need to keep it small.

**The Economic Argument**

In fact, all of the previous arguments boil down to this one. Stability and reliability contribute to productivity and therefore profits. Security is just the limiting of potential losses, tangible and intangible, but virtually all financial at bottom, at least in a commercial enterprise.

This is the weakest argument against a user demanding privileged access. But it underlies all the policies, all the procedures. Management should bear it in mind but not beat people up with it.

**Deputies**

Having said all that, there are some cases where extending privileged access to a few people outside the systems group will actually further all these goals. Former sysadmins, system programmers, and very knowledgeable users will often request such access when they find that their own productivity is impaired by waiting for systems staff (often overworked) to do apparently simple tasks as root. But the manager must decide if total productivity, stability, or security will be improved (or at least not damaged), rather than just the life of one user. It’s seldom an easy call.

Sysadmin groups usually develop some kind of processes that help them act as a team and avoid mistakes. The simplest is change control — to be able to roll back erroneous edits to system files and to track who did what. Further development brings formal communication and work tracking (mailing lists, ticket systems), and later, formal project planning, scheduling, and review; still later, metrics of the work processes themselves.

When these processes exist, helpers must be deputized by being brought into the loop and made to follow the standard procedures. Some effort is involved in training them for this, which can be the source of sysadmin resistance to deputizing someone. Sometimes, the request to become an official deputy sysadmin only serves to point out a glaring lack of such processes. Managers should recognize this situation, take steps to create (or just write down) the processes, and then to see that the prospective deputy is properly trained.

If helpers are not formally deputized, but just given the passwords without a second thought, they may end up being a source of grief for the sysadmins and the other users, and access may have to be revoked: clearly a failure of management. But if this is done right, a good deputy will be recognized as an ally to the sysadmins, and a resource to all the users.
the webmaster

Password-Protecting Areas of Your Web Site

Recently I was talking with a client about restructuring their Web site to make quite a bit of the material off-limits to the random public, areas with sensitive information that only their partners would be able to explore. The obvious solution was to employ a password-protection scheme, but there are a number of ways to approach this, and I thought it’d be interesting to share my thinking and results with the :login: crowd.

Password Protection, Take One

The first and perhaps most obvious way to do this is to replace a page of information with a login FORM on a Web page, then feed what the user entered to a CGI script that compares the password against the official password for that area of the site.

The HTML for the form might look like:

```html
<form action=login.cgi method=post>
Enter Site Password:
<input type=password name=pw size=20>
<input type=submit value='log in'>
</form>
```

Then, if we're going to utilize a Perl solution and have something like the simple cgi-lib.pl library from Steven Briner (you can get a copy for yourself from your local CPAN archive), the script underlying this is as simple as:

```perl
#!/usr/bin/perl
push @INC, '/cgi-bin/';
require ('cgi-lib.pl');
$official = "unix";
$ReadParse;
$pass=$in("pw");
if ( $pass eq $official ) {
    print "Location: logged-in.html\n\n";
} else {
    print "Content-type: text/html\n\n";
    print "Password Failed.\n";
} exit 0;
```

Enter the correct password ("unix", as defined in the fourth line of the Perl script) and you’re in: The next page you’ll see is "logged-in.html".

This first version works, but it’s pretty darn rudimentary and has some drawbacks, not the least of which is that everyone has the same password. Obviously, without account names to go with it, this is security only in the most minimal of senses.

There’s another problem here too, one that’s a bit more subtle: Once you’ve logged in, the URLs you’re seeing in your Web browser are post-login URLs. If you were to send, say, the real-life URL of the page we’re talking about here – <http://www.intuitive.com/CGI/password/logged-in.html> – to your friends, they wouldn’t even need to worry about the password because they’d have effectively skipped right past it.

You’d be surprised how many sites use this kind of mechanism for password protection. One password for everyone just isn’t very good. Having a password facade can be good for some cases, but isn’t very secure.
A Second Try

Before we leave this area, however, an example of where this kind of login might be useful is when you have individual accounts and want to display different information based on the type of account. In this case, the wrinkle is that you need to ask for both an account and password pair on the FORM (a trivial change), extract both in the CGI script, and then compare them against a file of defined account/password pairs. In this case, let’s create a file “pwfile” that contains two lines

    taylor:unix:partner
    guest:guest:guest

that define the account name, password, and level of access granted.

The second version of the Perl login script is a bit more complex. First, the subroutine that does all the work, reading and parsing the account file:

```
sub Matches
{
    local ($given_name, $given_pass) = @_;
    open(PASSWORDS, "pwfile") or die "can’t open pwfile;"
    while ($line = <PASSWORDS>) {
        chomp($line);
        ($name, $pass, $accesslevel) = split(":", $line);
        if ( $name eq $given_name) {
            if ( $pass eq $given_pass) {
                # success!
                return $accesslevel;
            } else {
                return nil;
                # wrong pw
            }
        }
    }
    close(PASSWORDS);
    return nil;    # no match on acct
}
```

Here’s where Perl is a winner: The `split()` routine automatically breaks up the line of information at the ‘:’ separator, returning each of the three values into its own mnemonic variable. Then the conditional tests are easy to code.

The main program needs to be modified to take advantage of the level of access that we can now grant:

```
#!/usr/bin/perl
push (@INC, "cgi-bin");
require ("cgi-lib.pl");
&ReadParse;    # read and parse arguments
$name='\$in('name')';
$pass='\$in('pw')';
if ($($access = &Matches($name, $pass)) ne nil) {
    print "Location: $access.html\n\n";
} else {
    print "Content-type: text/html\n\n";
    print "Account/Password pair failed.\n";
}
exit 0;
```

Do you see what’s happening here? If user “taylor” logs in successfully, he’ll be dropped onto the Web page “partner.html,” whereas if the guest user logs in, she’ll start out on the “guest.html” page.
With the passwords encrypted, it's a lot harder for hackers to reverse-engineer and sneak in if they manage to snag a copy of this information!

This is a cool way to control access to a site and simultaneously offer multiple levels of access. It still suffers from some of the limitations of the earlier solution, of course. If you see "guest.html," you might well guess "partner.html" was another possible Web page, and poof, you're in!

A Third Solution: Let the Server Do the Work
Yet another solution, one that offers more security, is to let the Apache Web Server do the work through the .htpasswd facility. In essence, you create a simple password file containing names and encrypted passwords, then enable your Web server to look for the file. Once set up, any access to any files within the protected folder must automatically be validated by forcing the user to enter a login/password pair.

To create the password file, I use a simple Perl script that I cobbled together called makepasswd:

```
#!/usr/bin/perl
print "\nMake htpasswd Account Entry...\n\n";
print "User name : ";
chomp($user = <STDIN>);
print "Password : ";
chomp($passwd = <STDIN>);
rand($$!time);
$saltchars=(a..z,A..Z,0..9,.,'/');
$salt.=saltchars[int(rand($$saltchars+1))];
$salt.=saltchars[int(rand($$saltchars+1))];
$passwdcrypt = crypt($passwd,$salt);
print \nAdd the following to the htpasswd file:\n\n";
print \t$user:$passwdcrypt\n\n";
exit 0;
```

As you can see, it does the work of encrypting the password and then displays exactly the information you'll need to add to the new .htpasswd file. A file duplicating the two accounts shown above would look like this:

```
taylor:z6K5hUNhVr1A
guest:SKUI6t2uiDOOs
```

With the passwords encrypted, it's a lot harder for hackers to reverse-engineer and sneak in if they manage to snag a copy of this information!

The only other step is to change the httpd.conf file so that the server knows to look for the password file in the directory. This is done by adding the boldface lines below to the file:

```
<VirtualHost www.intuitive.com>
ServerAdmin webmaster@intuitive.com
DocumentRoot /web
ServerName www.intuitive.com
<Location /cgi/password/private>
AuthName /web/cgi/password/private
AuthType basic
AuthUserFile /web/cgi/password/private/.htpasswd
Require valid-user
Allow From All
</Location>
</VirtualHost>
```

Now we're rocking! Any access to any of the information in the specified folder (AuthName) requires the user to log in to the server correctly, with the dialog box (as
shown in Figure 1) popped up and the account/password information compared against the contents of the AuthUserFile as shown above.

This is very cool and professional looking, being able to hide your information behind this kind of password protection. The downside is that you don’t get much control over the presentation and appearance of the box, whereas in the previous approaches you could build a login page that looked very consistent with the rest of the site’s appearance.

**Merging These Together**

Unfortunately, there’s no easy way to include additional fields in the htpasswd file data (which would be ideal), so instead you’re stuck having either to give out different password-protected URLs for different classes of users (for example, yourhost.com/partners/ and yourhost.com/guest/) or try a hybrid solution.

Let’s talk about the latter for a sec. It turns out that once you’ve logged in to a Web server with the htpasswd-prompted solution, for the duration of that session you now have an additional environment variable that you’re carrying around with you: REMOTE_USER. It’ll contain the name half of the name/password information required to log in.

With that in your toolbox, you could then have an index.cgi script, for example, that looks up the user in a second access-level file (keyed on the REMOTE_USER information), then presents a page based on that information. It’s a simple subset of what we’ve already seen; we don’t even need to worry about any CGI argument parsing.

The first part is the replacement for the match routine:

```perl
sub AccessLevel {
    local ($given_name) = @_;
    open(PASSWORDS, '../pwfile') or die "can’t open pwfile";
    while ($line = <PASSWORD>) {
        chomp($line);
        ($name, $pass, $accesslevel) = split(':', $line);
        if ($name eq $given_name) {
            return $accesslevel;
        }
    }
    close(PASSWORDS);
    return 'guest';
} # default access
```

If there isn’t a match in the file, it returns “guest” as the access level. Notice that I’m using the same file from the previous examples; it’s living one level up on the filesystem (../pwfile) but otherwise it’s as you’ve already seen.
Finally, here's the simple snippet that's the heart of the switch CGI script:

```perl
$name=SENV('REMOTE_USER');
$access=&AccessLevel($name);
print "Location: $access.html\n\n";
```

The variable `REMOTE_USER` contains the login name of the person who successfully signed in to the restricted area. If I just signed in as "taylor" (with the password "unix") then `REMOTE_USER` would be set to "taylor" automatically.

**Summary**

There's no perfect, graceful solution to password-protecting an area of a Web site with complete control, but this does give you a good idea of the different types of solutions and their trade-offs.

You can try out all these different solutions online and experience them for yourself: <http://www.intuitive.com/CGI/password/>.

---

**java performance**

In a previous column we looked at performance issues with Java I/O. Another aspect of I/O that needs to be mentioned is the cost of data formatting.

Consider first a couple of C examples, ones that output lines of the form:

```
The maximum weight is 100 lbs.
```

The first example simply prints this string repetitively:

```c
#include <stdio.h>
int main()
{
    const long N = 100000L;
    long i;
    for (i = 1; i <= N; i++)
        printf("The maximum weight is 100 lbs.\n");
    return 0;
}
```

The second uses `printf()` to format the weight and units values:

```c
#include <stdio.h>
int main()
{
    const long N = 100000L;
    long i;
    int w = 100,
    char* u = "lbs."
;
for (i = 1; i <= N; i++)
    printf("The maximum weight is \d %s\n", w, u);
return 0;
}

The second program runs around 25 percent slower than the first because of the formatting overhead. Formatting is clearly a useful feature, but it's worth knowing what the costs are.

With Java, a similar issue arises. To determine how expensive formatting is, we can write a series of programs. The first prints a string over and over:

```java
public class format1 {
    public static void main(String args[])
    {
        final int N = 25000;
        for (int i = 1; i <= N; i++) {
            String s = "The maximum weight is 100 lbs.\n";
            System.out.print(s);
        }
    }
}
```

The second uses the string concatenation operator (+) to format values of variables:

```java
public class format2 {
    public static void main(String args[])
    {
        final int N = 25000;

        int w = 100;
        String u = "lbs."
        for (int i = 1; i <= N; i++) {
            String s = "The maximum weight is \" + w + " + u + \"\n";
            System.out.print(s);
        }
    }
}
```

This code is slower than the first example because of the extra costs of converting integer values like "w" to strings, and because of costs in concatenating strings.

The third example uses the Text.MessageFormat facility:

```java
import java.text.*;
public class format3 {
    public static void main(String args[])
    {
        final int N = 25000;
        MessageFormat f = new MessageFormat("The maximum weight is {0} {1}\n");

        int w = 100;
        String u = "lbs."
        Object vals[] = new Object[2];
        vals[0] = new Integer(w);
        vals[1] = u;
        for (int i = 1; i <= N; i++) {
            String s = f.format(vals);
```
A format object of type MessageFormat is created and then used. This approach is desirable if you're trying to write Java applications that work in international contexts.

```java
System.out.print(s);
}
}
}

The idea here is that a format object of type MessageFormat is created and then used. Placeholders like "(0)" in the format are replaced with object values passed to the format() method. This approach is desirable if you're trying to write Java applications that work in international contexts; the message format can be read from a resource bundle (a group of resources that is keyed off of a locale) and then applied.

The final example is similar to the previous one, but it creates a message format on the fly:

```java
import java.text.*;
public class format4 {
    public static void main(String args[]) {
        final int N = 25000;
        String f = "The maximum weight is (0) (1)\n";
        int w = 100;
        String u = "lbs.";
        Object vals[] = new Object[2];
        vals[0] = new Integer(w);
        vals[1] = u;
        for (int i = 1; i <= N; i++) {
            String s = MessageFormat.format(f, vals);
            System.out.print(s);
        }
    }
}
```

In the previous example the message format was created once and then applied repetitively, allowing for some optimizations to be done. In this example, the format is passed in as a raw string each time. This approach is simpler but slower.

The programs produce identical output. The running times are:

<table>
<thead>
<tr>
<th>Format</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>format1</td>
<td>1.3 (seconds)</td>
</tr>
<tr>
<td>format2</td>
<td>1.9</td>
</tr>
<tr>
<td>format3</td>
<td>5.5</td>
</tr>
<tr>
<td>format4</td>
<td>8.0</td>
</tr>
</tbody>
</table>

If you're trying to tune I/O performance in a Java application, the area of data formatting may be worth looking at.
using java

Applets and Animation

A popular use of Java since its inception has been the creation of applets. Applets are the “other” way of running Java programs; they do not have a main method because the browser inside which they run has one instead. Applets are also special in the sense that they are “panels,” whereas Java programs that are not applets do not extend java.applet.Applet.

This article focuses on one of the more interesting and popular aspects of writing applets—animation. This typically means taking an object (text and/or graphics) and moving it on the screen. We will write a Java program that displays a “banner” in a browser. The code will include some aspects of programming with the AWT, and I’ll provide pointers on how to write applets. This applet also demonstrates the use of “threads” to do animation.

The AWT and the JDK1.1 Event Model

Let’s start with some fundamentals about applets. (This is a vast topic, so I won’t treat it very rigorously here.)

The graphics model is one of the key elements of the Abstract Windowing Toolkit (AWT), and applets are essentially panels that are part of the AWT. Applets are often used in interactive applications that run inside browsers, and so they rely on a powerful event model in the JDK to support the interactive capability. The event model is also basic to building GUIs (another potential use of applets).

The event model has changed considerably since JDK1.0. Although I know of no applications currently running under JDK1.0, they may still exist. The JDK1.0 event model was a “containment” model in which the events passed through the entire component hierarchy, and the programmer could control which components handled the event. The JDK1.1 event model is known as a “delegation” model. In this model, event sources and listeners are created, and components register themselves with the various event listeners. The event listeners handle the events generated from registered components. We will not deal with the containment model from JDK1.0 because the subsequent releases from JDK1.1 onward use the delegation model.

Applets

Applets are “panels”: The applet window has to be capable of containing other components such as text areas, buttons, and lists. The java.awt.Panel is the simplest class that can do this and be a top-level window.

One useful way to look at an applet is that it is defined by its context, such as a Web browser or appletviewer. Managing an applet means that the user must override methods such as when to start and what to do when another Web page is visited. The user must override these methods because java.applet.Applet defines them with empty bodies. These are the only methods that the user is permitted to override. However, the user is permitted to override those that Panel permits.

Example

This applet is an example of the use of a thread to do animation. The animation consists of moving some text across a background. This will also serve to introduce some new elements of the AWT, which I’ll discuss as we come across them.

Let’s start with the animation thread:

```
import java.awt.*;
import java.applet.*;
```
If the thread is runnable, then set it to null. The Java virtual machine will eventually notice that the thread created has no references to it and will flush it out of the system.

Import all relevant packages.

```java
public class Animation extends Applet
```

Animation is a subclass of applet and so is an applet itself.

```java
private Banner b;
```

Banner is the class that actually does the animation. We'll study that after we have looked at an applet that uses it.

```java
private Thread animationThread;
```

The thread that does the animation.

```java
public void init()
```

The init method that all applets must define.

```java
resize(Integer.parseInt(getParameter("width")),
        Integer.parseInt(getParameter("height")));
```

Read the parameters by the HTML file and set the size of the applet.

```java
b = new Banner("I Love Austin");
```

Create an instance of the banner class. The argument is an instance of String. This is the string that will move across the screen.

```java
add(b);
```

This is a new method. We will look at it in more detail later, but the effect of this is to tell the AWT system that b is to be contained within the applet. Remember that the class java.applet.Applet is a subclass of Panel, and panels can contain other AWT components.

```java
public void start()
```

The start method.

```java
if (animationThread == null)
    {
        animationThread = new Thread(b);
        animationThread.start();
    }
```

If no thread has been created, create it with an instance of Runnable as target. b is also an AWT component and is runnable.

```java
else if (animationThread.isAlive())
    animationThread.resume();
```

If it has been created and is runnable, then continue it from where it was, not from the start.

```java
public void stop()
```

The stop method.

```java
if (animationThread != null && animationThread.isAlive())
    animationThread = null
```

If the thread is runnable, then set it to null. The Java virtual machine will eventually notice that the thread created has no references to it and will flush it out of the system.

```java
public void paint(Graphics g)
```

```java
{
    super.paint(g);
}
```

Now let's introduce the Banner class:

```java
class Banner extends Component implements Runnable
```
Notice that Banner is a subclass of Component (which is an AWT component) because it extends Component, and it is runnable because it implements the interface Runnable. As you will remember, this means that instances of the class Banner can be supplied as targets when threads are created. This is exactly what we did in the applet previously.

Now for the various fields of the class Banner.

private String bannerString;

This is the banner that is dragged across the box.

private int boxw, boxh;

The size of the box across which the banner is dragged.

private Color fgColor;

The color of the font used for drawing the banner.

private Color bgColor;

The background color of the box on which the banner string will be drawn.

private Font bannerFont = new Font("Helvetica", Font.PLAIN, 36);

The font used for the banner string. We made it large... just like in real life.

private int[] X, Y;

An array of positions that are the starting points for drawing the banner string. The banner is first drawn at the coordinate (X[0], Y[0]), then it is moved by drawing at location (X[1], Y[1]), then at (X[2], Y[2]), and so on.

private int bannex, bannery;

The location at which the banner is currently drawn. This will move one of the members of the array X, Y.

Now that we have the Banner class we need to do a few more things. We must provide a body to the run method. Remember that run is part of the interface Runnable, and any class implementing Runnable must also provide a body for the run method. This is the method that will be called when an instance of the class Banner is passed as the target of the thread:

    public void run() {

The run method has two parts. The first part does some initialization. The second part does the actual animation. I'll discuss the local variables as we come across them.

    FontMetrics fm = getFontMetrics(bannerFont);

The class FontMetrics contains information about the font family. We need that to properly position the banner within this box.

    Dimension d = getSize();

This is the size of the component: its width and height.

    boxw = d.width;
    boxh = d.height;

boxw and boxh are fields of this class.

    delta = 1;

The amount (in pixels) by which the banner is moved to the left at each step. We choose to move the string by one pixel in each new step. This is a matter of trial and error and experience. A value that is too large will make it jerky. A very small value will make it computationally expensive. This example isn't very expensive, so we choose a small value.

April 1999 ;login:
\[ sw = \text{fm.stringWidth(bannerString)}; \]

Compute the width of this string in this font.

\[ n = sw/\text{delta}; \]

The number of steps it takes to draw the string across the box – this is just the size of the box divided by the size of each step.

\[ X = \text{new int}[n]; \]
\[ Y = \text{new int}[n]; \]

Create arrays for holding the location of the first character of the banner. Each pair of elements \((X[i], Y[i])\) specifies a location of the first character of the banner string.

\[ X[0] = sw * (\text{boxw} - sw)/2; \]
\[ Y[0] = (\text{boxh} + \text{fm.getAscent()})/2; \]

Set the zeroth element.

\[ \text{for (int } i = 1; i < n; i++) \{
X[i] = X[i - 1] - \text{delta};
Y[i] = Y[i - 1];
\}\]

The \(Y\) coordinate doesn't change, but the \(X\) coordinate is moved over to the left. Remember that we are just initializing things here; no animation is being done. We are, however, inside the thread. It is possible to move this initialization phase outside of run to the time when the banner is first created, but we chose not to do it that way.

\[ \text{while (true)} \]

Now we start the actual animation. This loop never ends, so we see the banner go round and round for ever. The loop just consists of setting the current location to a new position and then redrawing the screen – nothing complicated.

\[ \text{for (int } i = 0; i < X.\text{length}; i++) \]

For each element of the array (could be \(X\) or \(Y\) here; they are both the same length)

\[ \text{bannerX} = X[i]; \]
\[ \text{bannerY} = Y[i]; \]

Set the current location.

\[ \text{repaint();} \]

Repaint this component. repaint is a very important method. It has the effect of clearing the background and redrawing whatever needs to be redrawn. That is, it clears the background and calls the paint method. This is just what we need. When we change the position of the banner string, we want to rub it out at its previous location and draw it at the new location. This is just what repaint does.

\[ \text{try} \]
\[ \{ \]
\[ \text{Thread.sleep(100L);} \]
\[ \} \]
\[ \text{catch(InterruptedException ex) \{} \]

Just sleep for a while between steps in the animation – to slow things down a little.

\[ \text{try} \]
\[ \{ \]
\[ \text{Thread.sleep(2000L);} \]
\[ \} \]
\[ \text{catch(InterruptedException ex) \{} \]

After going through the entire animation, stop for a while before starting again.
Finally the all-important paint method must be provided:

```java
public void paint(Graphics g)
{
  Color c = g.getColor();
  Font f = getFont();

  Save the values of the current font and current color. This is not really necessary, but I
  have put it in just to show typical programming constructs. If you need to change col-
  ors and fonts multiple times, you will have to save old values if you need them later.
  Dimension d = getSize();

  The size of the box; this should be the same as boxw and boxh that we set previously.
  We could use those just as well.
  g.setColor(bgColor);
  g.fillRect(0, 0, d.width, d.height);

  Set the color and fill this box with that color. Remember that bgColor is black, so this
  will give us the black background we need.
  g.setColor(fgColor);
  g.setFont(bannerFont);
  g.drawString(bannerString, bannerx, bannery);

  Change the color again for drawing the banner, set the font in which the banner will be
  drawn, and then draw the banner string at the current location.
  g.setFont(f);
  g.setColor(c);
```

Strictly unnecessary, but you can reset the old values this way.

**Running Applets**

In order to view this applet we must now create an HTML file that has the name of the
applet. This is how a browser identifies which Java class contains the applet that is to be run. The following is the HTML file for this applet. You can call this file
animation.html or anything else with the same extension.

```html
<HTML>
<HEAD>
<TITLE>Animation Applet</TITLE>
</HEAD>
<BODY>
<H2>Animation Applet</H2>
<CENTER>
<APPLET CODEBASE="." CODE='Animation.class' name=animation width=400
  height=300>
  <PARAM NAME=banner VALUE="I Love Austin"></APPLET>
</CENTER>
<br>
</BODY>
</HTML>
```

As you can see, this is a very simple HTML file that will display a banner that says “I
Love Austin.” (It’s supposed to be a nice place.)

**Conclusion**

I have demonstrated by example the use of applets to do simple animation. In the
process, we gained some insight into the power of the AWT and the use of threads. The
Java environment is truly a rich one for writing powerful applications, be they console-
based or applets. In subsequent articles we will look at inter-applet synchroniza-
import java.applet.*;
biometrics: untruths and the truth

by Dario Forte

Dario Forte is an Italian system administrator. He is CCSE and CCSA CheckPoint Certified and is a USENIX-CSSI individual member. In Italy he moderated the independent forum of Windows NT Security and CheckPoint Firewall-1.

dario.forte@galactica.it

This is a journey through the new authentication mechanisms.

At a time when there is talk of “certainty of access,” of authentication of the users of a service, biometrics offers a solution.

Introduction

Biometric information – patterns of unique physiological and behavioral traits – can be used to authenticate access to given critical resources. Any organization may have strategic resources requiring protection. These can range from so-called sensitive data (data covered by laws concerning rights to privacy, for example), to mission-critical financial or commercial data, to military or legal information.

In addition to protecting data, an authentication mechanism based on biometrics can permit selective (and therefore sufficiently safe) access to given rooms or structures for which there is an intrusion risk.

System Components

A biometric authentication device is made up of three components:

- A database of biometric data. As you would expect, this is a large store of physiological and/or behavioral data. The stored information is compared with the input given at the time of access.

- Input procedures and devices. These are the systems (biometric readers, means for carrying the information, etc.) that connect the would-be guest with the validation system.

- Output and graphical interfaces. This is the front end. It is used to enter and display part of the access data and to obtain responses from the system.

Types of Biometric Data

We have described the components of a biometric authentication system. Now we shall explain the types of physiological and behavioral information that can be authenticated.

Possible physiological data include:

- retina prints
- fingerprints and palm prints
- voice prints
- keyboard input measurements
- iris recognition

At the moment, some feel that recognition of the retina is adequate both from the point of view of safety and, importantly, from that of the bandwidth required for an on-line transaction. It has been calculated that a network of devices for biometric authentication will take up about 32 Kb/sec. If a system’s available bandwidth is 128 Kb/sec, biometric authentication alone would eat up 25 percent of the total. Enough bandwidth must be reserved for other services, ranging from electronic mail to videoconferencing.

For fingerprint recognition, Compaq Computer Corporation is offering its Fingerprint Identification Technology at under one hundred dollars. Guaranteed to be compatible
with Compaq Deskpro, Armada PCs, and Professional Workstations, it is currently in
the testing phase for security applications in the Windows NT environment, and expe-
riements are being carried out on domain access as a replacement for conventional pass-
words. The fingerprint reader is placed near the video terminal and linked to a serial
port. It can be integrated with SmartCard use.

In addition to fingerprint scanning, voice recognition, and, in some military applica-
tions, dynamic measurements of character entry via a keyboard, biometric recognition
and the iris. The iris of the eye has a unique and visible
structure which is not currently possible to duplicate. It has been ascertained that the
human iris can identify an individual as accurately as his DNA. What is more, the iris is
stable throughout an individual’s lifetime.

Iris recognition is considered to be “just short of infallible,” definitely more foolproof
than fingerprint recognition. The pattern of the iris can be compared with the informa-
tion contained in an IrisCode database, with over 266 options for each record. The
scanning method is definitely one of the most transparent, since no physical contact
with the scanner is required.

The most crucial operation in iris recognition is the scan for the record in the database.
Strange as it may seem, the best results are achieved with a black-and-white camera.
According to IrisScan, the developer of iris recognition technology, black-and-white
scanning eliminates the possibility of incorrect recognition due to such factors as narc-
ocic or prescription drug use or colored contact lenses.

**Reliability**

What risk of error is there when using biometrics to control accesses? Many believe
the risk to be infinitesimal. Others are concerned, not so much about possible counter-
feiting of the physiological data of an individual, as about error on the part of the
scanners.

Scanner manufacturers deny the imputation. At the Sicur 98 meeting in Madrid,
Norberto Cartagena, sales manager of Ultra Scan of Miami, Florida, a firm that has
been active in the United States for over ten years, stated that at least for fingerprints,
the scanners currently available are reliable. Cartagena does not deny the need to opti-
imize some of their features; however, he sees this as part of the normal product up-
dating roadmap.

**Biometrics and the Internet**

An interesting step forward in integration between biometric devices and information
systems linked to the Internet has been made by iNTELITRAK Technologies Inc. Their
CITADEL GateKeeper has recently received security certification from the ICSA (for-
merly NCSA). The objective is to enable authentication of users of the Internet, an
intranet, or an extranet, not by password or other such standard means, but by voice-
pattern recognition.

CITADEL GateKeeper works as follows:

- The remote user links up with the authentication service via IP network or by tele-
phone.
- Once the link has been established, she follows the authentication instructions. (It is
worth noting that the voice input may be provided through a Sound Blaster-compatible
microphone.)
Gatekeeper carries out an analysis of the voice, comparing it with its authentication database, which can also interact with, among other things, X509V3 digital certificates. If a match is found, the system permits access to the information structures. If not, it follows an administrator-defined procedure to report an intrusion attempt.

Combining biometrics with conventional authentication methods carried out by, for example, a firewall or RADIUS server reduces the success rate of sniffers to a minimum. The system is fairly simple to use and also to integrate. The only concern lies in the error rate of the biometric recognition method. However, when the scanning and voice-pattern sampling, as well as the voice recognition, are carried out at different frequencies, maximum granularity should be attained.

**Administration of Biometric Systems**

Security operators recommend that the biometric database be administered by the security manager rather than a database administrator and that remuneration be directly proportional to the type of strategic resource being protected. This ensures both optimal safeguarding of the operator and a sense of responsibility for the project.

**To Hash or Not to Hash?**

Hashing, or calculating a numerical value for an input, usually based on the length of the datum in question, is intended to ensure the integrity of the hashed number during transmission via a network. Generally speaking, cryptographic algorithms are used to generate these functions and to code them. Hashing functions are currently used by programs such as PGP. It has been asked recently whether hashing functions should or could be added to biometric databases. Most experts feel that methods such as iris recognition are sufficiently safe, in particular when combined with the use of SmartCards.

**Has the Time Come for Biometrics?**

I recently talked with Cyril G. Reif, Director of Industry Technology, Financial Services Industry, at Sun Microsystems. Reif, who manages world-level accounts for Sun, reported on some comments heard from people in the banking sector. "Although the banking world does not exclude future use of biometrics in ATMs [Automatic Teller Machines], it is somewhat doubtful about this possibility, basically for reasons of lack of flexibility of use. They may possibly be used in the future; however, systems based on a Java SmartCard and on X509 digital signatures, which are currently the standard, are thought to be sufficiently safe."

In a recent interview in Foster City, Stephen Schapp, Deputy Chairman, Emerging Electronic Payments, of Visa International, confirmed that it is possible to have electronic payment methods interact with biometric authentication devices. Schapp himself, however, expressed some doubt about the use of authentication based on fingerprints, at least for ATMs. He felt that the scanning, checking, and authentication procedures would today require too much of the bank WAN's bandwidth. On the other hand, Schapp felt that it would be possible to use retina scanning in the future, although the applications based on this type of method require optimization.

Pilot implementations of this type have already been started by Visa International in the framework of the now famous Visa Open Platform project, introducing a suite of financial services based on new-generation SmartCards. Visa has implemented a Java-language software layer between the operating system of the card and the applications.
This layer acts as a bonding agent between the components described above, enabling multiple uses of a single SmartCard in electronic commerce.

In the past two months I’ve been traveling around Europe seeking to increase my knowledge of biometric products. I’m torn between two scenarios: digital fingerprint applications and iris recognition. Fingerprint-recognition vendors such as Siemens, Digital Persona, and Compaq are pushing their products hard, but customers are wary of the possibility of scanning errors. Iris-recognition vendors such as Iriscan, Sensar, Olivetti, and WangGlobal offer a very interesting alternative, yet one that threatens excessively high costs for ATM implementation. My personal conclusion: we’ll need to wait another eight to ten months for a solid biometric system – a reasonable period in the IT world.

source code UNIX

Embedding Source Code UNIX in the Product

The Essence is source code. If you have it, destiny is in your own hands, bringing flexibility and self-directed control. If not, you may fall prey to the whims of a large corporation and have to plead for bug fixes, enhancements, and device drivers.

Let’s consider some of the advantages and disadvantages of using Source Code UNIX as the operating system controlling an embedded product. I’ll examine the issues, the choices, and some concrete successes.

An “embedded product” has a processor (CPU), some memory, and a controlling program to produce the required features and functionality. Household appliances these days are often embedded products. A typical refrigerator has sensors to monitor the environment, including temperature and humidity. Based on these and the user-settable parameters (user interface), the controlling program gives commands to fans, motors, heaters, and lights to achieve the desired results.

Mass-produced embedded products have characteristics and economies of scale that warrant custom software – possibly cast in stone (or silicon). Usually the software is relatively simple and unchanging. Minimizing the cost of goods, especially the hardware, is extremely important. When you are building a million cars, stereos, or dishwashers, you’ll choose to spend $500,000 on a custom ASIC (custom application-specific IC) if it will save a few bucks per unit.

The embedded products that this article discusses resemble general-purpose workstations. Medical-imaging systems, laser printers, gateway routers, and vision systems have powerful processors, many megabytes of memory, disks, communications channels, and user interfaces. Often such products require virtual memory and multitasking. For these kinds of systems, the prices and quantities sold have characteristics different from more common and abundant appliances. In most cases, a short development time is more important than a small cost reduction for the hardware. The case for Source Code UNIX is strong for these products.
Operating System Features

Let's look at the issues involved when choosing software for controlling an embedded system. You'll want to consider how much total development effort you can afford and how long the development can take. Most organizations are eager to get things working in the shortest amount of time using a modest engineering team. Leveraging an existing, appropriate operating system will save costly software-development time. Further, risks are reduced because you start with part of the project already completed.

A key phrase in the above paragraph is "appropriate operating system." It wouldn't make sense to use the sophisticated UNIX operating system to control a washing machine. You need to look at your requirements and choose an OS that covers almost all of the needs. But plan for future features and enhancements, because most successful products have follow-on versions that place more demands on the software. For example, maybe the initial version of your product operates in standalone mode. If it is eventually going to be connected to other devices, you should think about an operating system that has network protocol stacks. Similarly, if "release 1.0" has all processes in physical memory, but you can see a requirement to add many more tasks in the future, you might want to start with a virtual-memory, paging system. Here are some questions to ask:

- Do you need multitasking, priority scheduling, and overlapped CPU-I/O?
- Do you need separate, memory-protected address spaces?
- Do you need networking capabilities?
- Do you need realtime capabilities (hard or soft)?

All of the above are large undertakings. Do you have the time and resources to build them yourself?

Operating System Choices

Let's say that you are the head software engineer responsible for building the product. If you are in from the beginning, you'll have some say in the selection of a processor type, a bus technology, a memory system, and peripherals. For example, MIPS has the largest number of embedded processors in the field — mostly because of its price/performance advantage over expensive chips such as the Pentium. But choosing this processor has to be balanced with the software-development costs. If you develop on Pentiums, you may need a cross-compiling environment and will sometimes be faced with problems such as endianness. For modest I/O requirements, a PCI bus will suffice; otherwise, you'll need a proprietary design. It's best to stick with mainstream components unless there are compelling reasons to go with specialty items. While considering hardware, look at your software choices and balance everything with cost, development effort, and time to market. Don't forget the cost of testing and maintenance.

The main Source Code UNIX systems, Linux and the BSDs, cover the popular processors: Motorola 680X0, Pentium, SPARC, PowerPC, StrongARM, MIPS. The same is true for the commercial realtime operating systems. (I'll use the term RTOS for the commercial embedded operating systems below. Most have hard realtime scheduling priorities.) Your choice between Source Code UNIX and an RTOS will boil down to what features are provided and at what cost. Rarely is source code available for an RTOS at a modest price. Often, you'll have up-front costs and per-unit costs to use RTOS binaries in your product.

The primary reason to consider Source Code UNIX is self-directed control of software development. You can examine the code to see in detail what is going on and modify it
to achieve your objectives. If you are lucky, there will be an RTOS that closely meets your requirements – it might not require any changes. But if it lacks features or performance crucial to your product, you could be in trouble. The vendor might be willing to make custom changes for you, but that will cost you time and money. Depending on the size and nature of the modifications, your budget or schedule might be broken.

In contrast, if you have source code, you always have the option of making enhancements yourself. If the system is misbehaving, you can track down the problem with your own resources instead of waiting for the vendor to get to it.

A unified environment for development and the embedded system enhances productivity. Any time you can carry out a large portion of software development and testing on your workstation, you improve the development cycle. The value of familiar compilers and tools should not be underestimated. While waiting for your hardware engineers to build your custom components, you may be able to simulate the equipment on a general-purpose workstation.

**Examples of Embedded-Systems Projects**

**A Video-Recorder Product**

In late 1995, as head of software in a tiny startup company, I had the job of designing and implementing the software to run a disk-based, high-end, video-recorder product. The combined audio/video data rate required storing or retrieving at about 30MB/second. Commodity disks of the day could only sustain a rate of about 5MB/second. Clearly we had to stripe the data across multiple disks. We also required redundancy against a single disk failure and achieved it with RAID3. We homed in on the mainstream technology of SCSI disks, PCI buses, and Intel Pentiums.

The naive hardware guys thought that NT would be the ideal software to run our product, but they had been brainwashed by a large corporate marketing organization, and they really didn't understand the issues. Given that we were going to build our own video card and motherboard (containing 20 SCSI buses, 3 PCI buses, Ethernet, and a high-bandwidth memory system with its own XOR engine), I saw that we needed source code to achieve our ambitious goals. The complexity of the product required a multitasking, virtual-memory, demand-paged operating system. We needed network protocols, a graphical user interface, and control over process scheduling. I knew we would require many enhancements in the SCSI subsystem for error recovery and hot swapability. Source Code UNIX was a great fit for the problem. Linux was a candidate, but I ended up choosing FreeBSD because of the better SCSI-driver support and because I had experience with BSD UNIX dating back to the late 1970s.

We leveraged many parts of FreeBSD source-code UNIX – many of which are not available in an RTOS. We had to get into the kernel to tweak PCI-bridge code, customize the serial driver, and make a handful of changes to boot-up code. We put much effort into improving the SCSI driver for our RAID system – the generic improvements were given back to the UNIX community. Our GUI leveraged X11 and Tcl/Tk. The product can be remotely upgraded with CVSUP (a software package for distributing and updating source trees from a master CVS repository; see <http://www.freebsd.org/handbook/>). Think of this as an elaborate EEPROM upgrade, such as you would do for a flashable modem.

We came up with a clever file-conversion mechanism using a lazy evaluation mechanism that leverages the UNIX vnode infrastructure. By accessing virtual files in the name space, implicit color-space conversions and file wrappers are invoked. See the code in FreeBSD kernels under `sys/miscfs` for examples. We served network files to
UNIX, Macs, and PCs using NFS and Samba. We wrote numerous custom drivers for
our hardware (temperature sensors, XOR engine, LED displays, etc.). We had existing
drivers as reference guides, so extending the code to match, for example, the DEC
21143 Ethernet chip was easy. We built the hardware and software from scratch with a
team of 10 in 14 months. I can't imagine completing such a project in a reasonable
amount of time without source code.

A Laser-Printer Product

A few of us are currently improving the performance of a medium-priced laser printer.
It needs a full-featured operating system to support tasks like interpreting and printing
PostScript, PCL, and rendering images, graphics, and other material. Printers have disks
for spooling, fonts, and paging. Interfaces to the printer include serial and parallel commu-
nications and Ethernet. The printer can receive jobs via FTP or spooling mecha-
nisms, or just by sending the bits. Status and control are available using Telnet or HTTP
Web interfaces. Printers of the current generation have huge virtual-memory needs. An
8.5 x 11 image with vertical resolution of 1200 dpi and horizontal resolution of 2400
dpi has about 256 mega pixels. Using a byte for each of cyan, magenta, yellow, and
black planes (CMYK) requires 1GB to hold an image. Then multiply that by double-
sided, multiple-copy, and n-up printing. Don't forget that page rasters have to be
retained in case of an error such as a page jam. I think you can see that printers have
more to their operating systems than it may first appear. We're using an older Source
Code UNIX–like system now, but plan to move the next generation MIPS R5000 prod-
uct to an OpenBSD base.

I'd like to point out that one of the remaining inefficiencies of the printer product is
TCP throughput. This is one of the few pieces for which we do not have source code.
The associated vendor is not very interested in improving his microcode.

An Image-Processing System

In the late 1970s, we were awarded a contract to build an image-processing system for
the Defense Mapping Agency. It would include a flatbed scanner, stereoscopic, high-
resolution displays, and adjacent floating-point processors. The commodity processors
of the time were PDP-11s, and the mainstream languages were assembler and Fortran,
under the DEC RSX11M operating system. We convinced those in power to go out on a
limb by using C with UNIX. We argued that we could deliver a more feature-rich sys-
tem sooner if we used a systems high-level language (C) and a Source Code UNIX.
Having source code allowed me to write a set of high-speed file routines (which were
given back to the community) to achieve the required I/O bandwidth. Many of the
same concepts are now mainstream in McKusick's Fast File System, which is the stan-
dard UNIX File System (UFS).

As a part of the image-processing system, we had a high-speed A/D converter connect-
ed to a PDP-11. The general belief was that even though UNIX had all of the right aux-
iliary processing tools for the collected A/D data, we would need a "realtime" kernel to
interface to the device. We had to spend only a couple of days to write a device driver
that sampled the data at interrupt time and buffered it. Ten lines of UNIX code were
changed. No samples were lost, and UNIX happily ran on the same computer.

Other Commercial Products Using Embedded Source Code UNIX

Cobalt Networks Qube 2 is a low-cost server appliance that provides Internet connec-
tivity, email, Web publishing, and other Web and network file services. Running Linux
on a 250MHz MIPS, it allows developers to add Web-server-based applications. See <http://www.cobaltmicro.com>.

The Whistle InterJet, using FreeBSD, provides email, Web access, and Web publishing for a small office. Whistle has been an active partner with the Open Source community, using and contributing to the Open Source projects. See <http://www.whistle.com>.

**When Not to Use UNIX**

Some RTOS vendors have extremely good products. Their code might fit your problem like a glove. If your problem matches the features of an existing RTOS but doesn’t fit the UNIX model, your choice is easy. Remember, many of the commercial vendors have a long history of serving particular areas in the market. Their mature product could be what you should use.

At first, I was going to recommend against UNIX for the small, nonpaged systems, but a Source Code UNIX can be stripped down to fit many situations. PicoBSD demonstrates that there are advantages for using UNIX even in tiny environments; it is a one-floppy version of FreeBSD, which in its different variations allows you to have secure dialup access, a small diskless router, or even a dial-in server. And all this on only one standard 1.44MB floppy. It runs on a minimum 386SX CPU with 8MB of RAM (no HDD required!). See <http://www.freebsd.org/~picobsd/picobsd.html>. Also see Linux tiny projects, below.

I would say that UNIX is a waste if you have only a single process. If you have no resources to control, you don’t need an operating system, but as soon as you want to have multitasking and process scheduling, the advantages begin accumulating.

**Conclusion**

Building good software in an embedded product is hard. For some applications, a commercial RTOS will fit the problem, but there are a large set of applications where a binary-only operating system will get in the way. Developers who have had UNIX source code know the tradeoffs. Programmers who have never had access to an open-source operating system might not appreciate the advantages. Peter Neumann[1] sums it up nicely: “The potential benefits of robust open-source software are worthy of considerable collaborative effort.” Give it a try.

**Reference**


**Resources for Source Code Unix**

You may feel that some consulting help or commercial support would best fit your needs. You have a number of options, including hiring consultants such as those found at:

<http://www.freebsd.org/commercial/consulting_bycat.html>
<http://www.openbsd.org/support.html>
<http://www.linux.org/business/index.html>
<http://metalab.unc.edu/DPH/HowTo/Consultants-HOWTO.html>

You could turn to companies that specialize in embedded Source Code UNIX:

Cygnus is a leader in Open Source–based software development tools, mission-critical support, and custom engineering for the embedded-systems market. 
<http://www.cygnyus.com>

---

PicoBSD demonstrates that there are advantages for using UNIX even in tiny environments; it is a one-floppy version of FreeBSD, which in its different variations allows you to have secure dialup access, a small diskless router, or even a dial-in server.
Boulder Labs specializes in high-bandwidth embedded UNIX systems. 
<http://www.boulderlabs.com>

RTMX O/S is based on OpenBSD; they add realtime scheduling, high-resolution timers, and contiguous file support. Complete source code for the RTMX O/S extensions and drivers is available. They support a large set of processors including those listed at <http://www.rtmx.com>.

Linux Router Project: <http://www.linuxrouter.org>
Linux on one floppy disk: <www.toms.net/~toehser/rt>

**Resources For Commercial RTOS**

pSOSystem has realtime multitasking kernel and networking support for a large set of processors. <http://www.isi.com/Products/pSOS/index.html>

VxWorks, from Wind River Systems, is a realtime operating system with networking facilities. <http://www.wrs.com/products/html/vxworks.html>

Inferno is designed to be a complete solution for the embedded market, uniting operating-system functionality, networking, and security within a small-footprint OS platform. <http://www.lucent.com/inferno>

BeOS was designed as an operating system for processor-intensive multimedia and Internet applications. <http://www.be.com>

LynxOS is a scalable realtime operating system with UNIX/POSIX APIs. LynxOS looks and feels like UNIX but was developed with deterministic hard realtime response in mind. <http://www.lynx.com>

QNX is a realtime, extensible POSIX OS with a lean microkernel and a team of optional cooperating processes. This flexible architecture lets you scale QNX down for lean embedded systems or scale it out to create a virtual supercomputer orchestrating hundreds of processors. <http://www.qnx.com>
musings

By now, you are all aware that Microsoft has ported Office to Linux. I have used a Beta version of the port and can report that Word works just like it does under Windows, including all of its annoying features (such as underlining phrases “it” doesn’t understand).

Bill Gates, in another of his famous “about-face” speeches, announced that Linux was obviously the wave of the future, and that Microsoft would soon be releasing extensions that would make Linux more compatible with Microsoft products. The source code to these extensions, and the underlying APIs, would require a standard Microsoft nondisclosure agreement to view.

Alas, I can no longer get away from using Office. Many of my friends and colleagues already consider Microsoft formats the lingua franca of the Internet and send me Word documents as email and schedules in Excel spreadsheets, and expect me to produce seminars in PowerPoint. I had been returning the documents and asking for something that I can read in vi, but Office for Linux makes it looks like dodging the issue will no longer be possible.

April fool. While it is true that Sun, HP, and Silicon Graphics announced that they would support a Linux port to their hardware, I don’t expect that Microsoft will offer a port of Office until it is too late to do them any good.

**Hard Facts**

Hardware vendors sell hardware. The operating systems help them to sell that hardware, which, you must admit, will just sit there uselessly without it. UNIX hardware vendors have traditionally used their operating systems as a means of differentiating their product offerings, when all we wanted them to do was make the operating systems work as alike as possible. Vendors instead put a lot of time, money, and effort in doing exactly the opposite. Now perhaps we can get what we have been asking for since the 80s.

Of course, the wider acceptance of Linux could also be its death knell. As Linux becomes more commercialized, different vendors will begin to include extensions that will make their version less interoperable with other versions. Even if these extensions are open source, they can still lead to problems.

Take for example the current scheme in HP/UX 10 and 11 for shadowing the password file. Instead of having a single shadow file that includes the encrypted passwords as well as additional, useful information (e.g., password aging and account expiration date), the HP/UX system uses separate files for each user account. These files are organized by directory, with a different directory for each letter of the alphabet. Most people that I talk to at the HP World conferences do not use shadowing under HP as a result of this. Now imagine that HP decides to “extend” Linux to use this same, overly complex scheme. Yikes! There goes the single version of your account management tool.

While I have great hope for the future of Linux, I am concerned about commercialization. Will Linux programmers “sell out”? Will there be Linux wars, like the UNIX wars of the late 80s and early 90s? I certainly hope not.

**The Desktop**

And while Microsoft won’t be supporting Linux any time soon, I think that many of us would agree that Linux makes more sense on desktops than does NT. But there are some problems here, many in the area of the ability of mere mortals to manage a UNIX system.
AT&T made an attempt to put UNIX on desktops. They created a stripped-down version of UNIX without most of the tools you would use in ordinary script writing (this was pre-Perl and Tcl). The result was a total failure. The operating system was too costly for desktops (about as much as NT is now), too hard for the user to manage, and unmanageable without the standard set of UNIX tools. And besides, a stripped-down UNIX system was just not politically acceptable to the very people who already used and managed UNIX systems.

Linux and other versions of UNIX today are just as unmanageable. While I do like a lot of the tools that come with Red Hat 5.1, it took a lot of man page reading and good ol’ UNIX-style fudging around to get pppd working from my laptop. I would not ask a Windows 9x or NT user to do that. And I would certainly not ask my mom.

The very flexibility of UNIX defeats most schemes designed to make it more manage-able. Yet, something must be done if UNIX is ever to take over desktops. Apple is working on this right now. MacOS 10, rumored to be out this year, will have the Apple desktop and applications sitting above the NextOS (Mach) kernel. Terminal windows and the UNIX command line may be there (I hope), and there was also supposed to be a Java Virtual Machine. Will Apple succeed where others have failed? I don’t know, but I do wish them luck.

Devices
The other big edge that Microsoft has over Linux is in hardware support. They have this because they have lots of person-power to manage and include thousands of different device drivers. And any hardware vendor is willing to write a decent device driver for Windows (and maybe, just maybe, NT too). Sun Microsystems has been the outstanding exception so far.

The list of supported devices (and motherboards) for Windows is perhaps an order of magnitude larger than it is for Linux or any other version of UNIX. While Linux already runs on more different processors than any other operating system, its list of supported devices is much smaller. Have you tried to install something other than a Sound Blaster sound card in a Linux system? If Linux won’t make noise when the user plays games, it’s all over for the desktop market (even if I consider Jaz drive support more important).

This will be the first barrier that must be overcome for the broader acceptance of Linux or any version of UNIX. Easy installation of Linux relies on the right device drivers being present or easy-to-get-and-install. Already, installing most UNIX systems is much easier (barring device-driver problems) than NT, and about four times faster.

Of course, there is one other really nasty device problem — and again Microsoft really lords it over UNIX in this area — and that is in setting up the windowing system. I am sure most of us have been through this in installing X on a PC. You must choose the correct timings for the X-window server to work correctly, and this is usually a hit-or-miss operation. What’s worse, the instructions tell you that poor choice of timing values might damage your hardware. Yikes! This “feature” helps me to understand why so many of my UNIX friends run Windows on their notebooks.

It would be nice if Sun’s Jini would help here. Jini is an object-oriented interface to devices that makes it possible to just connect a device to a network and have it instantly available and able to communicate without any device-driver installation or configuration. Essentially, software (Java, but it could be other) queries the device for its interface, and the device provides its own device drivers.
Jini won't work for bus-installed devices. Bus-installed devices are designed to work as fast as possible (that is, at bus speeds designed to rival memory speeds), and I don't think that Jini will work for disk, CD-ROM, sound cards, and other devices where latency is critical. I may be wrong, but I doubt it.

The real problem in writing device drivers has been corraling the device vendors. Hardware vendors pretty much do as they please. We have IBM to thank first, and Intel more recently, for there being any compatibility between PC devices at all. There have been attempts to standardize device interfaces, but none have succeeded at the software level necessary for making writing device drivers easy. Just getting data sheets from some vendors has proven impossible — unless perhaps it is Microsoft asking.

None of these problems is insurmountable, especially considering the legions of programmers who are working on them.

So I have a favor to ask. I would like those of you who are working on Linux or the BSD versions to send me article proposals on work designed to solve these problems. I am also interested in articles about how each organization decides what new features or code to include in each release, articles about the performance and reliability of the chosen filesystems, and plans to make Linux or other UNIX versions more manageable for mere mortals. I will propose having a special issue about this in \textit{login}; this year based on the response to this plea.

Please don't send me completed articles. Send me a page or two that clearly shows me that you know what you want to write about, know your topic, and can write. Don't worry if you are not an excellent writer — USENIX has a wonderful copy editor who will help you. Let's see if we can use \textit{login} to advance the future of operating systems in general, and UNIX in particular.

And please do not send me Word or any other Office documents. My bit bucket may be bottomless, but why waste your time?
The following Reports are published in this column:

**Open Source – A Standards Success Story?**

**Is Linux the Future of POSIX?**

Our Standards Report Editor, Nicholas M. Stoughton, welcomes dialogue between this column and you, the readers. Please send your comments to: <nick@usenix.org>

---

**edited by Nicholas M. Stoughton**

USenIX Standards Liaison

<nick@usenix.org>

---

**Open Source – A Standards Success Story?**

Nicholas M. Stoughton <nick@usenix.org>

This year, the buzz is all about Open Source. The LISA ’98 keynote speech, by Eric Allman, was about Open Source. Then, of course, there were the now infamous Hallowe’en documents from Microsoft. The idea has been around for a long time, but Eric Raymond’s paper “The Cathedral and the Bazaar” really sparked the most recent formalization of the concept. Those of you who want to know more about the concept should look at <www.opensource.org>.

Open Source software is extremely successful, in general. Without standards, it would probably still succeed, but it would have a far more limited appeal. Just how many successful Open Source applications are there that don’t have a POSIX-style system as the base starting point (even if they have been successfully ported to non-POSIX systems)? However, developers of Open Source software still have to riddle their code with #ifdef’s to allow for the enormous varieties of systems that are in existence. If the standards community had really achieved everything it wanted to, far less of this conditional compilation would be needed. But at least the majority of systems behave in a largely similar fashion, and that really is thanks to standards. In “The Cathedral and the Bazaar,” Eric Raymond points out, “When your code is getting both better and simpler, that is when you know it’s right.” The fewer the differences between systems, the easier it is to achieve this goal.

The idea that standards such as POSIX make source portability easier is most relevant when the source itself is open. Typical closed or proprietary applications do not worry so much about standards; they are out to use all the proprietary interfaces they can, to squeeze every last drop of performance out of the systems they have targeted. Just how much easier has POSIX made life for a company such as Oracle? (I am not qualified to answer that question personally, but I’d be very interested to hear a response from you if you work for an organization like that.) But how much easier has POSIX made it for the adoption of BIND or Apache?

It is very interesting to look through some of these Open Source applications and see where there is a lack of standardization. Perhaps we in the standards business should take more notice of these. For example, which header files are needed, and in what order; where certain files are located; and how options are handled regularly cause problems. These are the areas we need to consider in the current revision of POSIX and The Open Group’s Single UNIX Specification.

And what of that project? Last September, representatives from The Open Group’s Base working group, the IEEE Portable Applications Standards Committee (PASC), and ISO/IEC JTC1/SC22/WG15 (POSIX) met in Austin, Texas, to agree that the POSIX standards needed revising and to decide to do the work together. Because of the location, the group has become known as the Austin Group. It has taken six months to settle the basic ground rules for how that group will operate, but finally we appear to be ready to start the real work. Projects will be sponsored in each of the organizations to produce a single common set of specifications – initially four, covering what is now in the two POSIX.1 and POSIX.2 standards. The four books planned are: the system APIs (currently
POSIX.1 and XSH); Commands and Utilities (POSIX.2 and XCU); a new common definitions book; and a separate guide to using the standards, to which the rationale will be moved. The next meeting is planned for early March, with subsequent meetings in Montreal in July and somewhere in Europe (probably Copenhagen) in the fall. The first drafts of these four books should be ready before the year 2000 rolls around.

All Austin Group meetings are open to anyone; there is no membership requirement. All the drafts are intended to be freely available via the Web for comment. By helping to get these standards correct, useful, and usable, we can ensure a bright future for Open Source developers, our real target audience.

Is Linux the Future of POSIX?
David Blackwood <dave@rc.gc.ca>


“The 1980s and the 1990s have seen an explosion of Information Technology tools, development environments and development languages. Machine code, assemblers, imperative programming languages, Fourth Generation languages, Visual development languages, each promise to give us the edge in performance, code quality, development speed, simplicity for users and much more. As each new language or technology comes along, it is often embraced with enthusiasm and intense interest by some segment of the industry. It is only after a number of projects and challenging experiences that practitioners realize that they are locking themselves into vendor-specific solutions that make the task of upgrading, porting, and training new people much more difficult than they had anticipated. Alternatively, they are working in a mature language or technology but need the expressive power and concepts introduced by newer technologies, but preserving their investment in existing systems. These are the challenges of language standardization in the 1990’s. New languages and concepts arrive that somehow a globalization and solidifying process to guarantee portable tools and systems to practitioners. New concepts must be introduced into mature languages so that mature systems can adopt and grow with new technologies. International character sets and location profiles must somehow be integrated. And a world consensus is needed. The presentation examined the challenges faced by language standardization and how the ISO/IEC language standards groups cope to guarantee the most up-to-date and portable languages that can be imagined. – And we do it by consensus.”

Present that evening were a number of members of the Linux community who do not normally attend OCUUUG meetings, preferring instead to attend their own Ottawa Carleton Linux User’s Group (OCLUG) meetings. However, at the suggestion of a few individuals that OCLUG would be receptive to an invitation to this presentation, one had been extended to them several weeks earlier.

Attendance was good and several Linux users stepped forward afterward to express their interest in becoming involved in standards activities including in the Canadian POSIX Working Group (CPWG, equivalent to the US WG15 TAG). It remains to be seen how many actually follow through; however, the high level of interest was encouraging.

Interestingly, the week previous to this happening, discussions were held at the USENIX LISA conference between member of the standards and Linux communities about bringing more Linux users and developers into the formal standards process at the U.S. national and international levels. The outcome of these discussions was also encouraging. USENIX is considering sponsoring two representatives from the Linux community to participate in the U.S. national standards process.

One possible measure of system maturity is the emergence of viable divergent implementations. UNIX reached this point with the release of 2BSD in 1978 and continues the tradition today despite the POSIX family of standards. Some blame the failure of POSIX to create a “one true UNIX” for the rise of Windows NT as a potential competitor in the high-end workstation and server market. I believe that, seeking to avoid the mistakes of UNIX vendors in the past, the Linux community is beginning to realize the value of standards, and hopefully the value of POSIX. I would only further encourage them to use the influence of their numbers to direct the future course of POSIX and not to create yet another competing alternative in an already fragmented marketplace.

While Linux continues to gain mind share as well as market share, it struggles with how to handle product differentiation at the commercial level and yet try to maintain a single source tree for developers. Support from more traditional UNIX vendors continues to grow with recent announcements from HP, IBM, and Compaq. They join Oracle, Corel, and a multitude of smaller software vendors in providing support for the platform.

Is Linux the future of POSIX? Quite possibly. At the very least the future of POSIX looks a lot brighter if it includes Linux than if it doesn’t. Sun, SCO, and you other UNIX vendors, are you listening?
the bookworm

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 lustrum. He owns neither a dog nor a cat.

<peter@pedant.com></p>

Chapter 3 (pp. 79-108) concerns the early history of software, but it is full of lacunae. These are not errors! I think that Ceruzzi decided what to include and what not to include. And this is obvious in the detailing of the hardware, too.

In software, most of the OS and language work of the last decades is ignored. Ada, Smalltalk, Icon, Tcl, and Perl are among the languages unmentioned. Chorus, Mach, and Linux are among the operating systems. CTSS is mentioned on pp. 155ff, but Corbat’s name never appears. We are told that C was derived from B, but neither BCPL nor its creator, Martin Richards, is to be found.

At least part of this is understandable: Ceruzzi is a curator at the National Air and Space Museum, and museums have always been worse at collecting things that aren’t “hard.”

Ceruzzi wrote an excellent book on computers from 1935 to 1945, Reckoners: The Prehistory of the Digital Computer, some years ago. Despite the “sins” of omission, Ceruzzi presents an outstanding history of much of the hardware and some of the software of the past half-century. I can’t wait for his next one.

Security

Denning (Information Warfare and Security) has produced a first-rate book. But I have a confession to make: I was a reader of the manuscript for Addison Wesley and am thanked in the “Preface.” If this makes my review invalid, I’m sorry. Furthermore, I am complimenting Denning’s work despite the fact that she is a strong proponent of key escrow, to which I am opposed.

Denning has produced a volume of great importance: not a book about how to set up password files or firewalls, but a sober, adult discussion of the dangers of what she calls “information warfare.”

Denning starts out by recounting a tale of five Dutch crackers who broke into US DoD computers in 1991. She then moves
on to her view of information warfare, going into real detail about offense, defense, and the value of such resources to the opposing sides. She limns this going from playgrounds to battlegrounds.

The next sections are fascinating to read against Bill Cheswick’s paper at SANE’98 in the Netherlands. Denning is far more sober than Cheswick, but the warfare/battlefield analogies of Denning are completely analogous to Cheswick’s moats, sieges, treachery, etc.

Because of the nature of information warfare, Denning delivers a lengthy section on methods of attack: Using open sources to spy on individuals, copyright infringement, deception, hoaxing, defamation, spam wars, insider abuse, wiretapping, packet sniffing, telecommunications fraud, and about two dozen other things are detailed.

Finally, Denning looks at defensive measures, ending up with national policies.

One of the very best aspects of this book is the manner in which the frequent anecdotes are employed to increase the salience of subjects under discussion.

At the very end of Information Warfare and Security, the author writes, “The connection between encryption policy and security is not simple, and may be vastly overstated. Security demands much more than encryption, and encryption deployment is affected by factors other than government policy.”

She’s right.

Image Manipulation

GIMP is the GNU Image Manipulation Program, a “free” version of Adobe’s Photoshop, designed to be used on a variety of UNIX and Linux platforms. There was an article about GIMP in the November 1997 Linux Journal; now there’s a respectable and beautifully illustrated book (The Artists’ Guide to the GIMP) for prospective users. Hammel deserves the thanks of all those who design and work with computerized images. In barely 300 pages, he manages to introduce GIMP and elucidate the majority of its features, with high-quality images to show exactly what the GIMP tools and filters do.

GIMP is another demonstration of the bazaar taking over from the cathedral. I hope it becomes as widely used as it deserves.

Practice Makes Perfect

I mentioned Software Tools at the beginning of this column. It is still one of the best and most useful books on the “tool” philosophy. But The Practice of Programming is an absolute must for anyone reading ;login:. I am writing this on the basis of a draft manuscript, but the book will be out by the time you read this. In nine sections (“Style,” “Algorithms and Data Structures,” “Design and Implementation,” “Interfaces,” “Debugging,” “Testing,” “Performance,” “Portability,” and “Notation”), Kernighan and Pike serve up a scintillating volume that covers the gamut of skills and problems encountered by programmers.

My favorite passage is in Section 5.2:

Oops! Something is badly wrong. My program crashed, or printed nonsense, or seems to be running forever. Now what?

Beginners have a tendency to blame the compiler, the library, or anything other than their own code.

Experienced programmers would love to do the same, but they know that, realistically, most problems are their own fault.

The book ends with a brief epilogue and a three-page appendix: “Collected Rules.” If I had my druthers, every high school and college student taking programming would be compelled to learn them.

Among the best:

Be clear.
Be accurate.
Keep records.
Test incrementally.
Use standard compilers.
Don’t assume ASCII.
Don’t assume English.

Thanks, Brian and Rob.

Note

I’ve gotten a heap of books on ATM and another pile on Networking/Internetworking. They take a lot of time to actually look at. I hope to devote all (or most) of the June column to a number of these.

Notice of Annual Meeting

The USENIX Association’s Annual Meeting with the membership and the Board of Directors will be held at the Monterey Conference Center, site of the 1999 USENIX Annual Technical Conference. The date, time, and location of the conference will be published on the USENIX Web site <http://www.usenix.org/whatsnew/> in mid-May and will also be posted to comp.org.usenix at that time. This is a great opportunity to get your questions answered and to make suggestions on how we might serve you better. Everyone is welcome!
Most organizations have their own share of local-area networks. Multiple networks (both local- and wide-area) using a variety of protocols may have been inherited as the result of a consolidation or merger. Whatever their origin, these networks need to play well together. Making them do so requires the network practitioner to make choices. These choices must maintain a balance among cost, reliability, achieving today's goals, remaining compatible with legacy architectures, and providing a migration path to yet-to-be-defined networking technologies. For many organizations, building a network from the ground up is not an option.

SNA and TCP/IP Enterprise Networking, offered as a handbook for network practitioners, is based on the fundamental premise that multiple protocols will coexist forever. The authors feel this book will assist the reader with implementing reliable, cost-effective networking solutions with a migration path to future technologies.

The book first exposes the reader to a brief history of the development of multi-protocol communications. A brief discussion of the basic philosophical differences between SNA and TCP follows. The authors also describe important SNA multi-protocol integration products and popular deployment methods. The book closes with a look at emerging solutions. The authors illustrate both the strengths and weaknesses of each technology.

While the book provides a balanced presentation, its primary view of the world is from an SNA perspective.

The TCP and SNA sections open with a set of functionally oriented tutorials. They introduce SNA terminology and acronyms, and present the what and why of a subject. Many of these tutorials walk you through the flow of an SNA process. A couple of these chapters introduce TCP/IP concepts and terminology. Other chapters present application programming interfaces (APIs) that may be used with either of the two network protocols.

Chapter 8, the last chapter of part I, summarizes the information presented in the first seven chapters. In many places it provides a side-by-side functional comparison of the two network architectures and clearly illustrates the similarities and differences in performance between them. The authors review the two protocols' usability and reliability characteristics, and they present approaches to and methods for converging the two protocols.

Following the tutorials is an examination of SNA's interoperability features. The book's second part, "SNA Interoperability Today," describes currently available solutions: multi-protocol routers, gateways, and other software. Part II opens with a description of today's popular SNA internetworking strategy's benefits and pitfalls. The next chapter examines why one might want to encapsulate SNA within TCP/IP and provides some alternatives to encapsulation. Another chapter presents an approach for using SNA as the transport for non-SNA protocols. Topics such as providing 3,270 terminals access to TCP/IP applications, and questions and considerations when selecting or implementing an SNA gateway, are covered in the last couple of chapters of part II.

Host (mainframe) systems and their SNA networks have not gone away. Some new mainframes are even being sold as "enterprise servers." Part II concludes with examining ways Web technology can be used to enhance, extend, and leverage legacy platforms and the SNA architecture. This platform-independent client/server technology is inexpensive, easy to use, and readily available for many platforms. Web browser technology may just turn out to be the great equalizer among platforms.

The third section of the book, "Emerging Solutions," discusses the impact of recent technology on SNA presentation services. It suggests that future SNA applications will be "Common User Access compliant client/server applications written in Java." In closing, the last chapter gazes into the protocol crystal ball, presenting one view of how distributed computing technology may generate the next revolution in protocols.

Overall, the authors offer a model for synthesizing a multi-vendor, multi-protocol network into a cohesive whole that can appeal to applications and the user as a single integrated network.
The John Lions Award
by Dr. Lucy Chubb
Senior Consultant, Softway Pty Ltd.; President, AUUG
<lcucy@softway.com.au>

We are pleased to announce a second contribution by USENIX of $6,000 to the Lions Award fund. Ted Dolatta and John Mashey designated that the funds received from the auction of the California UNIX license plate be contributed to the fund.

The John Lions Award for Research Work in Open Systems was instituted in 1997 to honor the leading role Lions played in bringing UNIX to Australia, in the formation of AUUG, and in promotion of the values held by the open systems community.

The winner of the inaugural Lions Student Award in 1997 was Jerry Vochtelo, a Ph.D. student at the University of New South Wales. His work involved implementing UNIX-like "rwx" file protection on Mungi Objects (Mungi is a single-address-space object-oriented operating system that is being developed at the University of NSW).

Last year's winner was Steve Blackburn of Australian National University in Canberra, for work involving the creation of an orthogonally persistent version of Java. (I've read that his work has attracted interest from some of the big computer manufacturers.)

Over the past few years, I've heard it asked a number of times whether there is anything interesting going on in the operating system/open systems area. I believe there is. It's just that people aren't seeing it. This award should both encourage good work in the area and publicize the good work that's happening. Please visit <http://www.auug.org.au/lions/> for more information on the John Lions Award.

New Staff at USENIX

The small (average under 5'4") but energetic staff at USENIX has welcomed several additions lately. We'd like to introduce you to:

Gale Berkowitz. Before coming to the USENIX Association, Gale was on the faculty at the University of California, San Francisco. While her professional training is in public health research and epidemiology, she has long been interested in innovations in technology. In spite of her own experience with UNIX in the early '80s (which drove her to purchase her first Macintosh six months later) and her brother-in-law's opinion that UNIX was a gift from the devil (though both admitted that it has improved in measurably since then), she accepted her new position as Deputy Executive Director of USENIX without hesitation. As the DED, Gale will manage the day-to-day operations of the Executive Office, finances, and the Good Works and student programs, and will provide support to SAGE. She can also sometimes be found doing light housework around the office.

Bleu Castaieda. USENIX Administrative Assistant Bleu comes to us with a background in retail/customer service. She says she moved to San Francisco to study film production at the Academy of Art, but her apparent urge to visit every coffee shop in the world makes one wonder . . .

Cami Edwards. Southern California born and raised, USENIX Administrative Assistant Cami graduated with a B.A. in history from Cal State Long Beach. She's been on the Internet for close to five years and professes a fascination with all things UNIX and geeky. She can often be found dancing to techno or house.

Jane-Ellen Long. JE has combined computers and publishing throughout her working life, although her heart remains in Victorian England. She came to us from the University of California Press, where she served as Director of...
Information Systems. In her younger years she worked variously as production editor, copy editor, typesetting shop manager, even association conference manager. How old is she? You figure it out: she still has a punched card for a PDP-10. As Publications Director at USENIX, JE manages the Web site and print and online production of the conference proceedings, and serves as Managing Editor of *login*.

Jennifer Radcliffe. Jennifer was drafted to work at USENIX by the new Publications Director, Jane-Elene Long, for whom she had worked at the University of California Press, doing computer support and Web design. Her time at USENIX is mostly filled with creating and editing pages for the USENIX Web site. Her time away from USENIX is filled with surfing, playing electric violin, and dyeing her hair purple.

**20 Years Ago in USENIX**

*by Peter H. Salus*

<peter@pedant.com>

The January 1979 (Santa Monica) meeting had been a success: 350 attendees. The June meeting was scheduled for Toronto. The planning was underway.

First, there was an announcement of the "UNIX Users Group Conference" (the long arm of the AT&T lawyers hadn’t reached to Queens Park or St. Joseph Street yet), to be held at the University of Toronto, "Wednesday, June 20, 1979 through Saturday, June 23, 1979," with "Registration: Tuesday evening, June 19, 1979, 7:00PM - 9:00PM."

Second, there was an announcement of the "Software Tools User Group Meeting," to be held "June 19, the day before the general UNIX users meeting."

I’ll write about the meeting(s) in June. But in many ways, 1979 was to be an important and eventful one for UNIX. Toward the end of 1978, an early version of 32V – the port to the VAX by Charlie Roberts’s group in Holmdel – made its way from New Jersey to California. The code at the CSGS immediately began working on turning it into 3BSD. At the same time, the group at Bell Labs were turning V6 and 32V into V7.

By the time of its release, V7 was a truly wonderful system. Less than two years after DEC announced the VAX, there was a 32-bit OS (32V) and a new and updated version of UNIX for the PDP-11. Moreover, it contained Steve Bourne’s new shell, as well as gresp, uucp, awk, lex, lint, etc. The V7 manual was also the first to be commercially published: I still have my copy of the Holt, Rinehart and Winston printing of 1979.

V7 also served as input to 3BSD. In fact, when it appeared, 3BSD was everything that Bill Joy had said he wanted Berkeley’s “product” to be.

Bill had said that he was tired of doing releases that were made up of utilities and decided he wanted to release systems. 32V and V7 (which Berkeley had gotten in January) were the material he worked with.

3BSD was a complete, bootable system. It had a bootblock at the beginning of the tape, so you could roll it onto raw hardware. 3BSD had a virtual-memory-based kernel, and all the utilities had been boot-ed across. Perhaps more importantly, if you wanted to run UNIX with a paging system, you had to run 3BSD.

One of the people who wanted to run UNIX was Brian Harvey. In January 1979, Brian went to Lincoln-Sudbury Regional High School and persuaded the school board to float a bond issue for computer equipment. Brian’s 15-year-olds ran UNIX on a PDP-11.

Also in early 1979, Jim Kulp at the IASAS in Luxenburg, Austria, bought a VAX/780 and ran 3BSD on it.

Commercialization had also begun. In 1977 there was Interactive Systems; in 1978, P.J. Plauger’s Whitesmiths compiler and Idris, the first UNIX clone. Now, in 1979, with UNIX looking forward to its 10th birthday, there came XENIX – a collaboration between Microsoft and the Santa Cruz Operation – the first UNIX implementation for the Intel 8086.
There was going to be a lot to talk about in Toronto.

USENIX Funding Helps Improve the US Patent Process for Software

by Cynthia Deno
USENIX Marketing Director
<cyndia@usenix.org>

Invention in software technology, as in other fields, builds on previous advances. Inventions may be patented, unless they can be shown to be duplicative of previous work. In these years of light-speed advance in software development, the US Patent and Trademark Office (USPTO) has received and granted a rapidly increasing number of patents for software-related inventions. But some patents are better deserved than others.

Unfortunately, there is significant and growing uncertainty and controversy about the right to use a variety of software-related technologies. Software engineers run the risk every day of inadvertently infringing a patent, even though the particular technique being used is old and familiar. Inventors run the risk of having ownership of their contributions assigned to others by the USPTO.

Such controversy and uncertainty translate into a great many dollars for patent royalties, patent litigation, and efforts to avoid the use of materials patented by others. Three major groups are involved, the software industry — companies that rely on software as well as those that produce it — the USPTO, and patent professionals, along with individual inventors.

The Software Patent Institute was formed to help improve the patent process. It provides seminars and online access to prior art relevant to software-related technology. SPI has created a large and useful database of source documents — conference proceedings, journal articles, computer science theses, computer manuals, etc. — that are not readily available elsewhere. The USPTO is an enthusiastic supporter of the not-for-profit SPI, and patent office staff are frequent users of the database.

The USENIX Association, as part of its “Good Works” Program, recently granted the Software Patent Institute $55,000. This funding comes at an especially important time for SPI: they have a large backlog of material to load into their database but did not have enough money to go forward. This grant, says SPI Founder Dr. Bernhard A. Galler, will enable the continued growth and improvement of this increasingly important database resource, which benefits the software community as a whole.

The SPI Database of Software Technologies is accessible online by the public without charge. Along with helping the USPTO issue valid patents in the software field, the database helps software developers avoid the cost of defending against frivolous or otherwise invalid patents, and patent applicants can more easily and inexpensively research their claim.

The USENIX Association is wholly supportive of SPI’s mission to catalogue and make accessible source documents and software prior art. USENIX also recognizes the enormous task involved in achieving a more complete database.

The main problems encountered by SPI in building its database are obtaining copyright permissions and then funding the work of scanning and converting to machine-readable form early publications which exist only on paper. SPI staff are very efficient: a monospaced dissertation in Courier typeface can be put on-line pretty quickly. Documents such as the ACM Guides, however, with their tiny type and heavily abbreviated bibliographic entries, pose a larger challenge, as do documents with a lot of non-textual material such as graphs, equations, figures, or code. Nonetheless, even these yield to technology, patience, and time, and the SPI database is growing to provide an ever broader collection of older computer science materials.

For more information about SPI, consult <www.spi.org>.

WEB SITE
<http://www.usenix.org>

MEMBERSHIP
Telephone: 510 528 8649
Email: <office@usenix.org>

PUBLICATIONS
Jane-Ellen Long
Telephone: 510 528 8649
Email: <jel@usenix.org>

USENIX SUPPORTING MEMBERS

Apunix Computer Services
Cirrus Technologies
Cisco Systems, Inc.
CyberSource Corporation
Deer Run Associates
Hewlett-Packard India Software Operations
Internet Security Systems, Inc.
Microsoft Research
NeoSoft, Inc.
New Riders Press

Nimrod AS
O’Reilly & Associates
Performance Computing
Questra Consulting
Sendmail, Inc.
TeamQuest Corporation
UUNET Technologies, Inc.
Windows NT Systems Magazine
WITSEC, Inc.
5th Conference on Object-Oriented Technologies and Systems
Coots '99
Monday-Friday, May 3-7, 1999
Town & Country Resort Hotel, San Diego, California

Tutorial Program  Monday and Tuesday, May 3-4, 1999
Patterns at Work
Frank Buschmann, Siemens AG
Writing Efficient C++ Programs
Stan Lippman, Consultant
Implementing CORBA Servers Using the Portable Object Adapter
Steve Vinoski, IONA Technologies
Introduction to Java Beans
Uwe Steinmueller, Siemens Microelectronics
The COM(+) Programming Model
Don Box, DeveloperMentor
Programming for the Jini™ Platform
Ken Arnold, Sun Microsystems, Jini™ Team

Patterns and Performance of Real-time Object Request Brokers
Douglas C. Schmidt, Washington University, St. Louis
Distributed Java: Building Collaborative Applications
Ron I. Resnick, Dialogs, Inc.
Advanced Principles of Object-Oriented Design in UML
Robert C. Martin, Object Mentor Inc.
JavaBean Components: Specification, Design and Test with Catalysis/UML
Desmond D’Souza, Platinum Technology
Building Scalable ODBMS Applications
Matt BenDaniel, Object Design, Inc.

Technical Program  Wednesday, May 5, 1999
Opening Session
Opening Remarks & Awards
Murthy Devarakonda, IBM T.J. Watson Research Center

Keynote Address
James Arthur Gosling, Ph.D.
Chief Scientist, Java Software; VP and Fellow, Sun Microsystems, Inc.
James Gosling is currently a VP & Fellow at Sun Microsystems. He has built satellite data acquisition systems, a multi-processer version of Unix, several compilers, mail systems and window managers. He has also built a WYSIWYG text editor, a constraint based drawing editor and a text editor called ‘Emacs’ for Unix systems. At Sun his early activity was as lead engineer of the NeWS window system. He did the original design of the Java programming language and implemented its original compiler and virtual machine. He received a BS in Computer Science from the University of Calgary, Canada in 1977. He received a PhD in Computer Science from Carnegie-Mellon University in 1983. His thesis was entitled “The Algebraic Manipulation of Constraints”.

Design Patterns
Chair: Steve Vinoski, IONA Technologies, Inc
Filters as a Language Support for Design Patterns in Object-Oriented Scripting Languages
Gustaf Neumann and Uwe Zdun, University of Essen, Germany
Performance Patterns: Automated Scenario Based ORB Performance Evaluation
Sridhar Nimmagadda, Chanaka Liyanarachchi, Douglas Niehaus, Anil Gopinath and Arvind Kaushal, University of Kansas
Object-Oriented Pattern-Based Parallel Programming with Automatically Generated Frameworks
Steve MacDonald, Duano Szafra, and Jonathan Schaeffer, University of Alberta, Canada

Runtime Issues
Chair: Yi-Min Wang, Microsoft Research
Intercepting and Instrumenting COM Applications
Galyn C. Hunt, Microsoft Research and Michael L. Scott, University of Rochester
Implementing Causal Logging Using OrbixWeb Interception
Chana Shiy Namprempro, Jeremy Sussman, and Keith Marzullo, University of California, San Diego
Quality of Service Aware Distributed Object Systems
Svend Frolund and Jari Koistinen, Hewlett-Packard Laboratories

Objects and Databases
Chair: Rajendra Raj, Morgan Stanley & Company
Resource Control for Java Database Extensions
Grzegorz Czajkowski, Tobias Mayr, Praveen Seshadri, and Thorsten von Eicken, Cornell University
Address Translation Strategies in the Texas Persistent Store
Sheetal V. Kakkad and Paul R. Wilson, University of Texas, Austin

For detailed tutorial descriptions, please go to: http://www.usenix.org/events/coots99
Technical Program  Thursday, May 6, 1999

Invited Talk
Supporting Privacy in a Distributed Environment
Barbara Liskov
Ford Professor of Engineering, Massachusetts Institute of Technology
When servlets and applets are used today, it is impossible to protect the privacy of information that must be shared with the mobile code for it to do its job. This talk will describe new ways of providing control over information flow that protects the privacy of shared information. The talk will also discuss JFlow, an extension to Java that supports information flow control and allows most run-time information flow checks to be avoided by static checking.

Optimization
Chair: Werner Vogels, Cornell University

JMAS: A Java-Based Mobile Actor System for Distributed Parallel Computing
Legrand L. Burge III and K. M. George, Oklahoma State University

Adaptation and Specialization for High Performance Mobile Agents
Dong Zhou and Karsten Schwam, Georgia Institute of Technology

Design Considerations and Performance Optimizations for Real-time ORBs
Aniruddha Gokhale, Lucent Bell Labs; Irfan Pyrazi, Carlos O’Ryan, Douglas Schmidt, Vishal Kachroo, Alexander Arulampal, and Nanbor Wang, Washington University, St. Louis

Programming in the Large
Chair: Joe Sventek, Hewlett-Packard Labs

The Application of Object-Oriented Design Techniques to the Evolution of the Architecture of a Large Legacy Software System
Jeff Mason and Emil Occhotta, Xilinx Inc.

Supporting Automatic Configuration of Component-Based Distributed Systems
Fabio Kon and Ray H. Campbell, University of Illinois at Urbana-Champaign

Automating Three Modes of Evolution for Object-Oriented Software Architectures
Lance Tokuda and Don Batory, University of Texas at Austin

Java
Chair: Ken Arnold, Sun Microsystems, Inc.

The Design and Implementation of Guaran
Alexandre Oliva and Luiz Eduardo Busato, Universidade Estadual de Campinas, Brazil

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

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

Advanced Topics Workshop  Friday, May 7, 1999  9:00 am - 5:00 pm

Validating the Composition/Execution of Component-Based Systems
CHAIR: Joe Sventek, Hewlett-Packard Labs
Position Papers due: March 1, 1999  Acceptance Notices Issued: March 20, 1999

Review the Program.  See the Quality.  Join us at COOTS’99.
http://www.usenix.org/events/coots99

“I learned more in four days than three months of reading newsgroups, books, and journals. Excellent source of information and a chance to mingle with the big names in the field.”

Bob Laferriere, GE Medical Systems, Inc.

Register now. On-line registration: http://www.usenix.org/events/coots99
Tutorial Program    Sunday - Tuesday, June 6-8, 1999

Sunday, June 6, 1999

Windows NT Internals  New
Jamie Hanrahan, Harrys Systems
UNIX Security Tools: Use and Comparison
Matt Bishop, University of California, Davis
Linux Systems Administration
Bryan C. Andregg, Red Hat Software, Inc.
Essential UNIX Programming
Richard Stevens, Consultant
Network Security Profiles: A Collection (hodgepodge) of Stuff
Jon Rechlis and Brad Johnson, SystemExperts Corp.
Learning Perl
Tom Christiansen, Consultant
Secure Communications Over Open Networks
Marcus J. Ranum, Network Flight Recorder, Inc.
Advanced Solaris System Administration Topics
Peter Baez Galvin, Corporate Technologies, Inc.

Monday, June 7, 1999

Inside the Linux Kernel  Updated
Stephen C. Tweedie, Red Hat Software
Administering Windows NT: A Course for UNIX People  New
Adeen Frisch, Exponential Consulting
Multithreading Programming in POSIX  New
Bil Lewis, Lambda Computer Science
UNIX Network Programming
W. Richard Stevens, Consultant
System and Network Performance Tuning
Marc Staveley, Consultant
CGI and WWW Programming in Perl
Tom Christiansen, Consultant
Hot New Topics in Modern System Administration  New
Gary Nemerth, University of Colorado, Boulder; Ned McClain, XOR Network Engineering

Tuesday, June 8, 1999

Solaris Architecture: Internals, Tools, Tips and Tidbits  New
Richard McDougall and James Mauro, Sun Microsystems, Inc.
Sendmail Configuration and Operation (Updated for Sendmail 8.9)
Eric Allman, Sendmail, Inc.
Linux on the Edge  New
PART I: Real-Time Applications in Real-Time Linux
Victor Yodalken, New Mexico Institute of Mining and Technology
PART II: How to Build a Beowulf: Assembling, Programming, and Using a Commodity Supercomputer
John Salmon and Daniel Savares, California Institute of Technology

High Availability—Getting and Maintaining It
Evan Marcus, Veritas Software, Inc.
Advanced Topics in Perl Programming  New
Tom Christiansen, Consultant
Configuring Cisco Routers on an IP Network  New
William LeFebvre, Group Sys Consulting
Windows NT and UNIX Integration: Problems and Solutions  New
Phil Cox, Networking Technology Solutions
Modern Security Systems for Intranets, Extranets, and the Internet  New

Extreme Linux Workshop    Wednesday - Thursday, June 9-10, 1999

Linux is already running on millions of desktops and servers around the world.

However, it has also made significant inroads in the high performance computing community. Extreme Linux systems are the creation of supercomputer-class computing and graphics systems by using commodity, off-the-shelf computers combined with high speed networking, and glued together with Linux.

Useful in a variety of problems, Extreme Linux systems can be built for 1/30th of the price of an equivalent “proprietary” supercomputer.

USENIX is sponsoring the Second Extreme Linux Workshop. Designed by the originators of the first successful Extreme Linux Workshop, held last year in Santa Fe, New Mexico, this expanded workshop will concentrate on the issues of Extreme Linux systems. The workshop will be limited to 130 people, in a setting which will encourage participation and discussion. This is not a “presentation conference”, but a true workshop, where issues will be brought forth for discussion and resolution. Seating is limited, and invitations to Extreme Linux implementors and kernel/networking implementors will be given preference.

The workshop will focus on the research issues associated with the use of Linux in high-performance and supercomputing applications, including:

- Linux-based Workstation Clusters
- Wide-Area Supercomputing and Distributed Systems
- Filesystems, Device Drivers, and Kernel Support
- High-Performance Network Interfaces
- Message Passing and Distributed Shared Memory
- Applications
- Numerical Libraries
- Programming Language Support
- Graphics and Visualization
- Performance Analysis

For more information, see:
http://www.usenix.org/events/ useunix99/brochure/extreme.html

For detailed tutorial descriptions, please go to:  http://www.usenix.org/events/ useunix99
1999 USENIX Annual Technical Conference

Sunday - Friday, June 6-11, 1999
Monterey Conference Center, Monterey, California

Technical Program

Joint Opening Session

Wednesday, June 9, 1999

Opening Remarks
Avi Rubin, AT&T Labs—Research

Keynote Address
John Ousterhout, CEO, Scriptics Corporation
Integration Applications: The Next Frontier in Programming

Referred Papers

Resource Management
Implementing Lottery Scheduling: Matching the Specializations in Traditional Schedulers
David Petrou, John W. Milford, and Garth A. Gibson, Carnegie Mellon University

Retrofitting Quality of Service into a Time-Sharing Operating System
John Bruno, Jose Brustoloni, Eran Gabber, Banu Ozden, Abraham Silberschatz, Lucent Technologies, Bell Laboratories

Adaptive Modem Connection Lifetimes
Fred Douglas and Tom Killian, AT&T Labs—Research

Invited Talks

IP Telephony—Protocols and Architectures
Melinda Shore, Nokia IP Telephony Division

Rapid developments in IP telephony have, over the period of just a few years, moved us from a situation in which there were no standards into one in which there are many, often conflicting, standards. Different standards bodies, such as the Internet Engineering Task Force (IETF), the European Telecommunications Standards Institute (ETSI), and the International Telecommunications Union Standardization Sector (ITU-T) have developed their own models of how telephone systems should be constructed on packet networks, and what the interfaces to public and private telephone networks should look like. This talk provides an overview of current and developing protocols for IP telephony, as well as of the architectures which they were designed to support. Particular attention will be given to the interconnection of packet-based telephone systems and traditional, circuit-based telephony.

Will There Be a Transition to IPv6?
Allison Mankin, USC/ISI; Guy Davies, Worldcom UUNET-UK

In January 1995, after several years of work on multiple candidates, the Internet Engineering Task Force began the development of the consensus next generation of the Internet Protocol. The driver was the exhaustion of IPv4 address space. Four years later, that address space is indeed very scarce, but, notwithstanding some notable activities, such as the 6Bone, there appears to be little transition to IPv6. Davies and Mankin will describe the current tradeoffs of subscribers, equipment vendors and ISPs. They will evaluate the stability of the Network Address Translator (NAT) solution for address scarcity, present some expectations about device and embedded system uses of IPv6, and generally cover the question of whether there will be a transition to IPv6.

The Microsoft Antitrust Case: A View from an Expert Witness
Edward W. Felten, Department of Computer Science, Princeton University

Edward Felten recently served as an expert witness in the Microsoft antitrust case, and as a consultant to the Department of Justice. Edward will talk about his interesting and educational experience. He will share some of the things he learned about how the law, economics, and the software industry are connected.

FREENIX

File Systems

Soft Updates: A Technique for Eliminating Most Synchronous Writes in the Fast Filesystem
Marshall Kirk McKusick, Author and Consultant; and Gregory R. Ganger, Carnegie-Mellon University

Implementation and Performance of a Transaction-Based Filesystem on FreeBSD
Jason Evans, UNIX Developer

The Global File System: A Shared Disk File System for *BSD and Linux
Kenneth Presten, Matthew O’Keefe, University of Minnesota, and John Lekashman, NASA Ames

Device Drivers

Distributing Device Drivers Outside of the Linux Kernel
Theodore Ts'o, MIT

Design and Implementation of Firewire Device Driver on FreeBSD
Katsuaki Kobayashi, Communication Research Laboratory

newconfig: An Dynamic-Configuration Framework for FreeBSD
Atsushi Furuta, Software Research Associates, Inc.; and Jun-ichi Itoh, Research Laboratory, Internet Initiative Japan Inc.

File Systems

The Vinum Volume Manager
Greg Lehey, Nan Yang Computer Services
Porting the Code Distributed File System to Windows 95
Peter J. Braam, Carnegie Mellon University; Michael J. Callahan, Stelios Computing, Inc.; and M. Satyanarayanan, Carnegie Mellon University

A Network File System Over HTTP Remote Access/Modification of Files and "files"
Oleg Kiselev
1999 USENIX Annual Technical Conference
Sunday - Friday, June 6-11, 1999
Monterey Conference Center, Monterey, California

Technical Program
Thursday, June 10, 1999

Invited Talks

Y2K: UNIX/Open System meets Real World IT Issues
Alan F. Nugent, Independent Consultant
To some, the Year 2000 problem is an overblown, reactionary, non-issue mushroomed by dinosaur COBOL programmers who have nowhere else to go and greedy consultants. To many it is frighteningly real and requires immediate remediaion. Like so many things, the answer lies somewhere in the middle and definitely in the eye of the beholder. The simple truth is: there are some electronically controlled devices that will function perfectly through the millennium change, while others will need to be fixed or retired. This talk will examine the many facets of the Y2K problem as it exists in the real world, some of the practices for remediation, potential consequences of action and inaction, and a retrospective of creative solutions born out of corporate America.

FREENIX

Security
A Future-Adaptable Password Scheme
Niels Provos, University of Michigan; and David Mazières, OpenBSD Developer

Cryptography in OpenBSD: An Overview
Thao de Raadt, Nicklas Hallqvist, OpenBSD Developer; Artur Grabowski, Ericsson Telecom AB;
Angelos D. Keromytis, University of Pennsylvania/OpenBSD; Niels Provos, University of Michigan

Minding Your Own Business: Platform for Privacy Preferences Project
Lorrie Faith Cranor, AT&T Labs—Research

Invited Talks

FREENIX

Networking
Trapeze/IP: TCP at Near-Gigabit Speeds
Andrew Gallatin, Jeff Chace and Kenneth Yoczum, Duke University

Managing Traffic with ALTQ
Kanjiro Cho, Sony Computer Science Laboratories, Inc.

Opening the Source Repository with Anonymous CVS
Chuck Cranor, AT&T Labs—Research; and Theo de Raadt, OpenBSD Developer

Tools and Platforms

Lightweight Structured Text Processing
Robert C. Miller and Brad A. Myers, Carnegie Mellon University

SBOX: Put CGI Scripts in a Box
Lincoln D. Stein, Cold Spring Harbor Laboratory

The MultiSpace: An Evolutionary Platform for Infrastructure Services
Steven D. Gribble, Matt Welsh, Eric A. Brewer, and David Culler, University of California at Berkeley

Web Servers

Web++: A System for Fast and Reliable Web Service
Radek Vangrallek, Yuri Breitbart, Lucent Technologies Bell Laboratories; Mehmet Sayal, Peter Scheuermann, Northwestern University

Efficient Support for P-HTTP in Cluster-Based Web Servers
Mohit Aron, Peter Druschel, and Willy Zwaenepoel, Rice University

Flash: An Efficient and Portable Web Server
Vivek Pai, Peter Druschel, and Willy Zwaenepoel, Rice University

Web Caching

NewsCache—A High Performance Cache Implementation for Usenet Text
Thomas Gschwind and Manfred Hauswirth, Technische Universität Wien

Reducing the Disk I/O of Web Proxy Server Caches
Carlos Maltzahn, University of Colorado Boulder; Kathy Richardson, Compaq Computer Corporation; Dirk Grunwald, University of Colorado Boulder

An Implementation Study of a Detection-Based Adaptive Block Replacement Scheme
Jongmoo Choi, Sam H. Nah, Sang Lyul Min, Yoockun Cho, Seoul National University

Caching

E-mail Bombs, Countermeasures, and the Langley Cyber Attack
Tim Bass, Consultant
The robustness of the Sendmail MTA program can be misused in numerous attack scenarios to create dangerously destructive SMTP e-mail bombs. These e-mail bombs are launched by readily available automated software tools which can easily crash chains of SMTP mail servers. SMTP mail relays can also be used covertly to distribute messages and files that could be seriously damaging to the integrity and brands of victims. This talk discusses SMTP mail-bombing techniques, automated attack tools, countermeasures, and "The Langley Cyber Attack."

The speaker, who was the Chief Scientist during the 1997 attack, will discuss the analysis of the cyber attack, graphs illustrating the attack volume, and a statistical e-mail bomb early warning system. Recent anti-spam enhancements to sendmail are compared to the e-mail bomb countermeasures and the "blackhole strategy" used in the Langley Cyber Attack.

Register now. On-line Registration: http://www.usenix.org/events/usenix99
Technical Program

**Friday, June 11, 1999**

### Refereed Papers

**Operating Systems Structure**

A Scalable and Explicit Event Delivery Mechanism for UNIX
Gaurav Banga, Network Appliance, Inc.; Jeffrey C. Mogul, Compaq Computer Corporation, Western Research Lab; Peter Druschel, Rice University

The Pebble Component-Based Operating System
Eran Gabber, John Bruno, Jose Brustolon, Ari Silberschatz, and Christopher Small, Lucent Technologies, Bell Laboratories

Linking Programs in a Single Address Space
Luke Deller, and Gernot Heiser, The University of NSW

**Storage Systems**

The Design and Implementation of DCC Device Driver for UNIX
Tycho Nightingale, Yiming Hu, and Qing Yang, University of Rhode Island

An Application-Aware Data Storage Model
Todd A. Anderson and James Griffioen, University of Kentucky

### Invited Talks

**Big Data and the Next Wave of InfraStress Problems, Solutions, Opportunities**
John R. Mashey, Silicon Graphics/Cray Research

Data storage is growing at a higher rate than ever before, and coupled with rapidly increasing demand for instant access, will cause great stress on both the physical and the human infrastructure of computing. System planners and administrators will soon face the interesting challenge of dealing with network and backup issues when office systems hold 100s of GB of disks, and larger servers reach 10s and 100s of TB and even PB. There will also be great opportunities in both research and commercial applications, but the problems must be understood, and solutions anticipated. This talk will give some examples, including some large customer problems that Silicon Graphics has been working on; and examine technology trends in storage capacities, access times, computer architectures, and bandwidths, to see what these portend over the next few years.

**What's Wrong with HTTP And Why It Doesn't Matter**
Jeffrey Mogul, Compaq Western Research Lab

HTTP quickly grew to become the dominant protocol on the Internet, but its maturation as a protocol design hasn't been as speedy. The HTTP/1.0 specification was written only after the protocol had been deployed, and the IETF working group chartered to design HTTP/1.1 took 4 years to produce a Draft Standard. What we have now is a useful but still seriously flawed protocol.

Jeffrey Mogul was one of the primary authors of HTTP/1.1. This talk will give his personal view of what is still wrong with HTTP, and what we can learn from these mistakes. These include fundamental conceptual errors (the lack of true extensibility, the inappropriate analogy to MIME, and the confusion around caching) and some other problems with the standardization effort. This talk will explain why he doesn't think these errors matter and how HTTP, flawed as it is, still solves problems. This talk will also describe why various efforts to extend or replace HTTP may not pay off.

**UNIX to Linux in Perspective**
Peter Salus, UNIX Historian

Born in 1969, UNIX grew, matured, morphed and was even cloned. Its maturation cycle created international standards as well as multiple for-profit and not-for-profit companies. It became the lingua franca of the computer research and development community. Today, the many variants of UNIX claim 30 million users worldwide.

UNIX was 22 when Linus Torvalds created Linux, a UNIX clone. By 1998, this clone had 5 million users in its own right. Earlier decades had seen successful UNIX strains arise, but as of today fewer than 5 major variants survive. This talk will briefly recap 1969-89, concentrating on the extolation of UNIX and its clones over the past 10 years.

### FREENIX

**Applications**

Berkeley DB
Mike Olson, Sleepycat Software

The FreeBSD Ports Collection
Satoshi Asami, University of California, Berkeley

Multilingual vi Clones: Past, Now and Future
Jun-ichiro Higino, Research Laboratory, Internet Initiative Japan Inc.; and Yoshitaka Tokugawa, WIDE project

**Kernel**

Improving Application Performance through Swap Compression
Raul Cervera, Toni Cortes and Yolanda Becerra, Universitat Politecnica de Catalunya

New Tricks for an Old Terminal Driver
Eric Fischer, University of Chicago

DENTS—A Server for the DNS Protocol
Todd Graham Lewis, MindSpring Enterprises, Inc.

### Works-in-Progress Session

Session Chair: Keith Smith, Harvard University

WORKS-IN-PROGRESS REPORTS (WIPs)

Do you have interesting work you would like to share, or a cool idea that is not yet ready to be published? The USENIX audience provides valuable discussion and feedback. Short, pithy, and fun, Works-in-Progress Reports (WIPs) introduce interesting new or ongoing work. We are particularly interested in presentation of student work. Prospective speakers should send a short one- or two-paragraph report, to Keith Smith at wips99@usenix.org.

There are a limited number of slots available for work-in-progress presentations. Proposals for WIP presentations will be accepted at the discretion of the WIP chair, with preference given to those that are received earliest.

Joint Closing Session:

The USENIX Quiz Show!

HOSTED BY ROB KOLSTAD

VISIT OUR WEB SITE: http://www.usenix.org/events/usenix99
**Workshop on Smartcard Technology**

**Monday-Tuesday, May 10-11, 1999**  
McCormick Place South, Chicago, Illinois

Sponsored by USENIX and Co-sponsored by CardTech/SecurTech

**First of Its Kind in North America**

The USENIX Workshop on Smart Card Technology brings together researchers and practitioners in the development and deployment of smart card systems and technologies. Join them for this unique opportunity to learn about what is possible today in smart card technology, and on the drawing boards for tomorrow.

**Advanced Technical Program**

Peer-reviewed papers and selected presenters from around the world cover the latest in smart card research and advanced development in:

- smart card hardware
- smart card software
- system issues
- strengths and weaknesses of smart cards
- smart cards’ role in operating systems
- smart cards as a base technology in cryptographic systems

**Held in conjunction with**

**The World’s Largest Card & Security Exhibition**

Attend the USENIX Workshop on Smart Card Technology and enjoy free admission to the CardTech/SecureTech ’99 Exhibition, May 12-14, 1999, in McCormick Place South, Chicago. For more details, go to [http://www.ctst.com](http://www.ctst.com)

---

**Technical Program**  
*Monday - Tuesday, May 10-11, 1999*

| Keynote Address | Fred Renner  
VP, Chief Technology Officer, Global Transaction Company |
|-----------------|----------------------------------------------------------|

**Design**

- Feasibility of the Smart Card in Silicon-On-Insulator (SOI) Technology  
  Amaury Neve, Dr. Denis Flandre, and Jean-Jacques Quispelier, Microelectronics Laboratory of Université Catholique de Louvain, Belgium
- Design Strategies for Tamper-Resistant Card Processors  
  Olivier Kämmerling and Markus D. Kuhn, University of Cambridge
- Which Security Policy for Multiplication Smart Cards?  
  Pierre Girard, Gemplus, R&D

**Ciphers**

- Efficient Block Ciphers for Smartcards  
  Vincent Rijmen, K.U. Leuven, Dept. ESAT, Belgium, and Joan Daemen, Proton World Internet!, Belgium
- PKCS#15 - A Cryptographic Token Information Format Standard  
  Magnus Nyström, RSA Laboratories
- Remotely Keyed Encryption Using Non-Encrypting Smart Cards  
  Stefan Lucks and Ruediger Weis, Universität Mannheim

**Authentication I**

- Smartcard Integration with Kerberos V5  
  Naomaru Itō and Peter Honeyman, CITI, The University of Michigan
- A Portable Solution for Mutual Authentication  
  Bastiaan Bakker, Lifeline Networks BV
- Software License Management with Smart Cards  
  Tuomas Aura, Helsinki University of Technology, and Dieter Gollmann, Microsoft Research

| Keynote Address | Dr. David Everett  
Technical Director, NatWest Group |
|-----------------|----------------------------------------------------------|

**Authentication II**

- Beyond Cryptographic Conditional Access  
  David M. Goldschlag and David W. Kravitz, Divx
- Providing Authentication to Messages Signed with a Smart Card in Hostile Environment  
  Tage Stabe-Kulå, Ronny Arild, and Per Harald Myrvang, University of Tromsö
- Authenticating Secure Tokens Using Slow Memory Access  
  John Kelsey and Bruce Schneier, Counterpane Systems

**Operating Systems**

- SCFS: A UNIX Filesystem for Smartcards  
  Naomaru Itō and Peter Honeyman, CITI, The University of Michigan
- Java Card Secure Object Sharing  
  Michael Montgomery and Kshearabdi Krishna, Schlumberger Austin Product Center
- Object Lifetimes in Java Card  
  Marcus Oestreich, IBM, Inc., and Kshearabdi Krishna, Schlumberger Austin Product Center
- A Personal Naming and Directory Service  
  Alain Macare and David Carrier, Gemplus Research Lab, France

**Threats**

- Investigations of Power Analysis Attacks on Smart Cards  
  Thomas R. Namerikawa, IBM, and Robert H. Sloan, University of Illinois at Chicago
- Risks and Potentials of Using EMV for Internet Payments  
  Et H. Van Herreweghen, IBM Zurich Research Laboratory, and Uta Wille Jelml, AG Information Systems, Zurich, Switzerland
- Breaking Up is Hard to Do: Modeling Security Threats for Smart Cards  
  Bruce Schneier, Counterpane Systems, and Adam Shostack, Netscape, Inc.

Visit our web site: [http://www.usenix.org/events/smartcard99](http://www.usenix.org/events/smartcard99)
SUBSCRIPTION ORDER FORM

PROCEEDINGS

Please enter my one-year subscription to the 1999 USENIX Conferences and Symposia Proceedings, which includes:

3rd Symposium on Operating Systems Design and Implementation – February 1999
1st Conference on Network Administration – April 1999
1st USENIX Workshop on Intrusion Detection and Network Monitoring – April 1999
5th Conference on Object-Oriented Technologies and Systems (COOTS) – May 1999
USENIX Workshop on Smartcard Technology – May 1999
3rd USENIX Windows NT Symposium – July 1999
2nd Large Installation System Administration of Windows NT Conference – July 1999
8th USENIX Security Symposium – August 1999
2nd Conference on Domain-Specific Languages – October 1999
2nd USENIX Symposium on Internet Technologies and Systems – October 1999
13th Systems Administration Conference (LISA ’99) – November 1999

NOTE: Corporate, Educational, and Supporting members of USENIX automatically receive a subscription to all proceedings published during their term of membership, which may overlap with this offer.

COST: USENIX members: $185.00* Subscription Cost
Non-members: $260.00 Calif. Sales Tax
Outside U.S.A. and Canada? Overseas Postage
Add $36.00 for surface postage.

TOTAL ENCLOSED

* The member price also applies to members of EurOpen National Groups, AUUG, and JUS
**If you are paying member price, please include member’s name and/or membership number

PAYMENT OPTIONS

☐ Check enclosed payable to USENIX Association
☐ Purchase order enclosed
☐ Please charge my: ☐ Visa ☐ MasterCard ☐ American Express

Account # ______________________________________________________________________ Exp. Date __________
Signature ______________________________________________________________________

Outside the U.S.A.? Please make your payment in U.S. currency by one of the following:
• Check - issued by a local branch of a U.S. bank
• Charge (Visa, MasterCard, American Express, or foreign equivalent)
• International postal money order

Name ______________________________________________________________________
Address ______________________________________________________________________
City ____________________________ State/Country ____________________________ Zip/Postal Code __________

Please mail this order form to: USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
510/528-8649
FAX 510/548-5738
office@usenix.org

This offer is good through December 31, 1999

The proceedings are shipped the week following the event.
Important Dates:
Extended abstracts due: May 25, 1999
Invited Talk Proposals due: May 25, 1999
Notification to authors: June 30, 1999
Final papers/Invited Talks due: September 1, 1999

Conference Organizers:
Program Chair:
David Parter, University of Wisconsin
Program Committee:
Eric Anderson, University of California, Berkeley
Strata Rose Chalup, VirtualNet Consulting
Xev Gittler, Goldman Sachs & Co.
Trent Hein, XOR Network Engineering, Inc.
Doug Kingston, Deutsche Bank
Tom Limoncelli, Lucent Technologies / Bell Labs
Bryan McDonald, Global Networking And Computing, Inc.
Adam Moskowitz
Ervi Nemeth, University of Colorado
Cat Okita, Earthworks
Josh Simon, Collective Technologies
Jeffrey A. Uphoff, Transmeta Corporation
Invited Talks Coordinators
Jennifer Katinsky, Silicon Graphics, Inc.
Phil Scarr, Global Networking and Computing, Inc.
Practicum Chair
Bruce Alan Wynn, Collective Technologies
The Guru Is In Coordinator
Lee Damon, Qualcomm, Inc.
Conference Advisor
Rob Kolstad, SANS Institute

The Only Conference By and For System Administrators
LISA, the Systems Administration Conference, is the largest and oldest conference for system administrators. The LISA experience is special because the entire program is assembled by veteran systems administrators who know first-hand the issues you face, and what factors are important in devising solutions. Participants are involved in all aspects of modern systems administration, including user support, workstation and server administration, network management, security and incident handling, and site-wide planning and design.

LISA offers the most comprehensive program for system administrators from sites of all sizes and at all levels of experience. The conference includes tutorials, refereed papers, invited talks, "Guru" and "Birds-of-a-Feather" sessions, and informal discussions with your peers and other experts in every area of Systems Administration.

Conference Topics
The Program Committee invites you to join the contributors to the LISA XIII conference. Submissions of refereed papers or other presentations which address any and all aspects of System Administration are acceptable. Here is a partial list of timely paper topics for potential authors:
Technology, Tools, and Techniques
- Innovative system administration tools and techniques
- Tips and tricks: new uses for old tools
- Distributed or automated system administration
- High availability and disaster recovery
- Scaling support of "open source" systems for servers and desktops
- Designing, selecting, scaling, integrating and managing "enterprise" computing services.
- Security (all aspects)
- Authentication systems
- Applications of tools, techniques and methods from other disciplines
- Integration of new networking technologies, protocols and applications
- Integration of emerging technologies
- Performance analysis and monitoring

Theory and Practice of System Administration
- Methodology, paradigms and models for system administration
- Analysis of "best practices" in systems administration
- Analysis and comparison of alternative systems for systems administration tasks
- Case studies
- Application of scientific methods to systems administration
- Metrics for systems administration

The "Soft Science" of System Administration
- System administration management issues
- Support strategies
- Effective hiring techniques
- Effective training techniques for system administration
- Budgeting, cost analyses and project planning/management techniques for systems administration
- The System Administrator's role in the organization

Tutorial Program
Gain mastery of complex techniques and technologies and you'll get immediate payoff within your organization. You can choose from up to 40 full- and half-day classes over three days. Whether you are a novice or senior systems administrator, you will be able to find a tutorial to meet your needs. Tutorials cover important topics such as: performance tuning, administering Windows NT, Perl, TCP/IP troubleshooting, security, networking, network services, sendmail, Samba, legal issues, and professional development.

Submitting A Tutorial Program Proposal
To provide the best possible tutorial offerings, USENIX continually solicits proposals for new tutorials. If you are interested in
presenting a tutorial at this or other USENIX conferences, please contact the tutorial coordinator:

Daniel V. Klein
Tel: 1.412.421.0285
Fax: 1.412.421.2332
Email: dok@usenix.org

Workshops
Up to three limited-attendance, special-topic workshops will be offered alongside the Tutorial Program at LISA '99. If you have suggestions for topics or the methodology of a workshop, please send them to the program chair by email to:
lisa99chair@usenix.org

Technical Sessions
Three days of technical sessions feature parallel tracks of refereed papers, invited talks, the Practicum track, and "The Guru is In" sessions for individual consultation with experts on specific topics. Refereed papers are published in the Proceedings (provided free to Technical Sessions attendees). Invited Talk and Practicum track materials are made available online.

The refereed papers will provide the latest on cutting-edge technologies. Refereed papers might be academic in nature, designed to advance the field of systems administration, or they may report practical solutions to specific problems. Papers that analyze problem areas and draw important conclusions from practical experience are especially welcome.

Cash Prizes
Cash prizes will be awarded at the conference for the best paper and for the best paper by a student. Prizes are for papers accepted to the refereed paper track.

Submitting an Invited Talk Track Proposal
If you have a topic of interest to systems administrators that is suitable for an invited talk, please submit a proposal to the Invited Talk coordinators. Please email your proposal to:

iti@usenix.org

Submitting a Practicum Track Proposal
The Practicum track includes a variety of presentation types and focuses on discussion of practical solutions to System Administration issues. If you have a topic that fits the Practicum track, please email a proposal to:
lisa99practicum@usenix.org

Works-in-Progress Reports
Do you have interesting work you would like to share, or a cool idea that is not yet ready to be published? The USENIX audience provides valuable discussion and feedback. We are particularly interested in presentation of student work. To schedule your short report, send email to:
lisawips@usenix.org

Birds-of-a-Feather Sessions
Birds-of-a-Feather sessions (BoFs) are very informal gatherings organized by attendees interested in a particular topic. BoFs are held Tuesday, Wednesday, and Thursday evenings. BoFs may be scheduled in advance by phoning the Conference Office at 1.949.588.8649 or via email to: conference@usenix.org. BoFs may also be scheduled at the conference.

How and Where to Submit a Paper to the Refereed Track
An extended abstract of two to five pages is required for the paper selection process. Full papers are not acceptable at this stage; if you send a full paper, you must also include an extended abstract.

Include appropriate references to establish that you are familiar with related work, and, where possible, provide detailed data to establish that you have a working implementation or measurement tool.

Submissions will be judged on the quality of the written submission, and whether or not the work advances the state-of-the-art of system administration. More detailed author instructions, including sample extended abstracts and final papers, will be available in March on the conference web site at:

http://www.usenix.org/events/lisa99

Note that LISA, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication, and that submitted papers not be previously or subsequently published elsewhere for a certain period of time. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned unread. All submissions are held in the highest confidence prior to publication in the conference proceedings, both as a matter of policy and as protected by the U.S. Copyright Act of 1976.

Every accepted paper must be presented at the conference by at least one author. Authors of an accepted paper must provide a final paper for publication in the conference proceedings. One author of each accepted paper receives complimentary technical session registration. Final papers are limited to 20 pages, including diagrams, figures and appendices. Complete instructions will be sent to authors of accepted papers.

To discuss potential submissions and for inquiries regarding the content of the conference program, contact any member of the program committee or the program chair:

David Parter
Email: lisa99chair@usenix.org
Tel: 1.608.262.0608

All submissions for LISA ’99 will be electronic. A web form for submissions will be available in March.

Program and Registration Information
Complete program and registration information will be available by September 1999 at the conference website at:

http://www.usenix.org/events/lisa99 The information will be printable from a PDF file.

If you would like to receive the program booklet in print, please request it at any time. Email your postal address to conference@usenix.org. Be sure to mention you would like to receive the LISA ’99 program.
ALL THE TOOLS AND APPLICATIONS YOU’LL NEED...
ABLE TO HANDLE TENS OF MILLIONS OF HITS PER DAY!

It's not a bird or a plane, it's the BSDI Internet Super Server 4.0! The "Super" platform of choice for thousands of Internet Service Providers (ISPs) and corporate Intranets. Recorded uptimes in excess of one year, all the Internet services, development tools and utilities you'll need to soar above your competition. Affordable price, superior support, and a 60-day money back guarantee make Internet Super Server your "Super Hero". Be sure you've got the most dependable platform in the Industry — Internet Super Server 4.0!

Call BSDI now and order Internet Super Server 4.0!
+1-800-286-2731
+1-719-593-9445 or
E-mail: info@bsdi.com
Visit our web site at: www.bsdi.com/products/internetbw

10% Discount To USENIX Members
Call 1-800-776-2734 with your USENIX number.
Addison-Wesley Brings you the BEST in Computing

Get solutions, tips, and practical advice from the experts in the field

Special
20% Discount*
for USENIX Association members

Save 20% on all Professional Computer and Engineering books from Addison-Wesley. Mention Discount Code 719LI 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!
http://www.awl.com/cseng/

*Offer expires on 12/31/99
Competence

I just love dealing with competent people and organizations. It’s truly a joy. Brief conversations like this are the highlight of my life these days:

“How do I fix my frazzlegadget?”

“Oh, just turn the montipeter gear three times counterclockwise.”

“Thanks!”

“Glad to be of service.”

Regrettably, I fear that the computer revolution (along with the TRILLION DOLLAR INTERNET OPPORTUNITY (tm)) has ruined it for me. No longer can I call my ISP for support and talk to someone with a clue. Instead, I must hang on hold listening to – in the case of my current ISP – whiny wailing violins playing some wretched piece of classical music that was emphatically not in my mother’s collection when I was growing up.

After five, maybe ten, minutes of this auditory torture, I get to speak to a Real Human Being (tm). Unfortunately, this human being’s greatest strength is patience with idiots (tm).

“Have you rebooted your computer?” is a typical cheerful question.

“Oh, I can’t help you with anything as technical as DNS, let’s fill out this service request form together.”

Sometimes it’s enough to make me puke.

Last month was interesting. “I am calling to report that your e-mail server is down and no mail is being delivered to it.”

“I am sorry you’re having trouble, sir. I can guarantee that we can fix that problem by . . . oh . . . let’s say next Tuesday, four days from now.”

“@#$%!@#$%.” And then I feel bad, because my mother taught me better than to swear at total strangers (as if swearing at your best friends was somehow a better thing).

I think that competence – or at least its perception – is one of the things that binds software people and system administrators (and their ilk) together into a sort of huge brotherhood (siblinghood? O where to stop . . .).

In fact, this competence might even be causing some of our mutual problems these days:

“Oh shoot. I need this garbage disposal on the net right NOW.”

“Don’t fret. I’ll call Sue; she can network anything.”

And, of course, Sue (a.k.a. Ms. Johnni-on-the-spot) is loathe to admit any potential path to failure. “Sure, I’ll stay up tonight and skip all my meals and my brother’s wedding tomorrow so that I can make up for your lack of planning! Thanks for this opportunity to serve you!”

I think this scenario, while exaggerated in a certain Dilbert-ish sense, nevertheless happens far too frequently in our industry. I think that it causes stress and creates a sort of hopeless set of ever-escalating expectations from which escape is impossible without changing companies – and I think that’s unhealthy.

On the other hand, it beats a cheery “Please reboot your computer, sir,” all to hell. Thanks to everyone that is maintaining high standards of service in this increasingly automated world. You make living and working worthwhile.
SPECIAL OFFER FOR USENIX MEMBERS ONLY!

Usenix and the IEEE Computer Society have an ongoing agreement that benefits you! *IEEE Concurrency* is available to Usenix members through technical partnership at a 49% discount.

*IEEE Concurrency* covers advances in computer architectures and programming languages that enable you to achieve new levels of system performance by distributing functions across networks and allowing different processors to operate concurrently on multiple aspects of a problem. Departments, tutorials, and feature articles in *IEEE Concurrency* give software and hardware perspectives on subjects as diverse as collaborative computing, real-time systems, mobile computing, database and transaction processing, heterogeneous computing, middleware, and visualization.

**Departments and columns keep you**

- **current**
  - Industry Spotlight
  - Global Broadcast
  - Applications Focus
  - Computing Trends
  - Product News

- **competitive**
  - Mobile Computing Trends
  - Object-Oriented Systems
  - Distributed Databases
  - Multimedia Applications
  - Internet Issues

- **informed**
  - Viewpoint
  - Interviews
  - Virtual Roundtable
  - Education Issues
  - Book Reviews

**Feature articles focus on topics like:**

- Actors and Agents
- Data Mining
- Distributed Databases
- Mobile Computing
- OO Technology
- Workflow
- And more!

**IEEE Concurrency—practical information you need to stay on the cutting edge**

**Don’t delay—order today!**

---

**RISK-FREE USENIX MEMBER Order Form**

- [ ] YES, SIGN ME UP!
  YES! I want to subscribe to a full year of *IEEE Concurrency* at the special Usenix rate of only $45!

  Use the individual member ID (required for this offer)

  Name

  Mailing Address

  City

  State/Country ZIP/Postal Code

  Charge to: [ ] Visa [ ] MasterCard [ ] American Express

  Charge-card number: ____________________________

  Expiration date: _______ ______

  If USA, please include billing address, zip code.

  Signature: ____________________________

100% MONEY-BACK GUARANTEE — If you are ever dissatisfied, you may cancel your subscription and receive 100% of your money back.

- [ ] Check enclosed

TO ORDER, send complete information and payment to:

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314

Fax for fastest service to 1-714-821-4641

Phone: 1-714-821-8380
E-mail: membership@computer.org
http://computer.org/concurrency
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 June 1, 1999, and August 3, 1999.