inside:

CONFERENCE REPORTS
USENIX 1998 Annual Technical Conference

SAGE NEWS & FEATURES
Taming the Certification Beast
Speakers Bureau
Effective Perl Programming

STANDARDS REPORTS
The Future of POSIX

BOOK REVIEWS
The Bookworm
Java Network Security
Sun Performance and Tuning
LDAP
Managing Mailing Lists

USENIX NEWS
Letter from the President
Summary of USENIX Board Meetings
CSRG BSD Releases Available
Twenty Years Ago in ;login:
USENIX Annual Awards
USACO Camp Completed

features:

Interview with L. Peter Deutsch
by Stig Hakvån

Source Code UNIX on the PC
by Bob Gray

The Security Model of JDK 1.2
by Dario Forte

Java Performance
by Glen McCluskey

Using Java
by Prithvi Rao

Musings
by Rik Farrow
# Upcoming Events

**1st International System Administration and Networking (SANE) Conference**
Organized by NLUUG, cosponsored by USENIX and Stichting NLNet

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

**12th Systems Administration Conference (LISA '98)**
Co-sponsored by USENIX and SAGE

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

**USENIX Workshop on Smartcard Technology**
Sponsored by USENIX and co-sponsored by CardTech/SecureTech

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 10-11/99</td>
<td>Chicago, IL</td>
<td>Scott Guthery &amp; Peter Honeyman</td>
</tr>
</tbody>
</table>

**USENIX Annual Technical Conference**

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 7-11/99</td>
<td>Monterey, CA</td>
<td>Avi Rubin, Program Chair, Clem Cole &amp; John Heidemann, IT Coordinators, Jordan Hubbard, Freenix Track Chair</td>
</tr>
</tbody>
</table>

**3rd Symposium on Operating Systems Design and Implementation**
Co-sponsored by ACM SIGOPS and IEEE TCOS

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

**3rd USENIX Windows NT Symposium**

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>July 12-16/99</td>
<td>Seattle, WA</td>
<td>Werner Vogels &amp; Stephen Walli</td>
</tr>
</tbody>
</table>

**1st Conference on Network Administration**
Co-sponsored by USENIX and SAGE

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program chair</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 7-9/99</td>
<td>Santa Clara, CA</td>
<td>David Williamson</td>
</tr>
</tbody>
</table>

**2nd Large Installation System Administration of Windows NT Conference (LISA-NT)**
Co-sponsored by USENIX and SAGE

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program co-chairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>July 12-16/99</td>
<td>Seattle, WA</td>
<td>Gerald Carter and Ralph Loura</td>
</tr>
</tbody>
</table>

**1st USENIX Workshop on Intrusion Detection & Network Monitoring**

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program chair</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 11-12/99</td>
<td>Santa Clara, CA</td>
<td>Marcus J. Ranum</td>
</tr>
</tbody>
</table>

**Eighth USENIX Security Symposium**

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program chair</th>
</tr>
</thead>
<tbody>
<tr>
<td>August 23-25, 1999</td>
<td>Washington, D.C.</td>
<td>Win Treen, Program Chair, Avi Rubin, IT Coordinator</td>
</tr>
</tbody>
</table>

**5th Conference on Object-Oriented Technologies and Systems (COTTS)**

<table>
<thead>
<tr>
<th>WHEN</th>
<th>WHERE</th>
<th>WHO program chair</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 3-7/99</td>
<td>San Diego, CA</td>
<td>Murthy V. Devarakonda</td>
</tr>
</tbody>
</table>

### Deadlines

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

<table>
<thead>
<tr>
<th>DEADLINES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Abstracts</td>
<td>Notification of Acceptance</td>
</tr>
<tr>
<td>Final Papers</td>
<td>Final Papers</td>
</tr>
<tr>
<td>November 6/98</td>
<td>Dec. 16/98</td>
</tr>
<tr>
<td>December 1/99</td>
<td>January 12/99</td>
</tr>
<tr>
<td>Final Papers</td>
<td>March 30/99</td>
</tr>
</tbody>
</table>

**USENIX Workshop on Smartcard Technology**

<table>
<thead>
<tr>
<th>DEADLINES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Abstracts</td>
<td>Notification to Authors</td>
</tr>
<tr>
<td>Final Papers</td>
<td>Final Papers</td>
</tr>
<tr>
<td>December 1/98</td>
<td>January 20/99</td>
</tr>
<tr>
<td>Final Papers</td>
<td>April 27/99</td>
</tr>
</tbody>
</table>

**USENIX Annual Technical Conference**

<table>
<thead>
<tr>
<th>DEADLINES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper Submissions</td>
<td>Notification to Authors</td>
</tr>
<tr>
<td>Final Papers</td>
<td>Final Papers</td>
</tr>
<tr>
<td>December 2/98</td>
<td>January 20/99</td>
</tr>
<tr>
<td>Final Papers</td>
<td>April 27/99</td>
</tr>
</tbody>
</table>

**3rd Symposium on Operating Systems Design and Implementation**

<table>
<thead>
<tr>
<th>DEADLINES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Notification to Authors</td>
<td>Final Papers</td>
</tr>
<tr>
<td>October 13/98</td>
<td>January 6/99</td>
</tr>
</tbody>
</table>

**3rd USENIX Windows NT Symposium**

<table>
<thead>
<tr>
<th>DEADLINES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper Submissions</td>
<td>Notification to Authors</td>
</tr>
<tr>
<td>Final Papers</td>
<td>Final Papers</td>
</tr>
<tr>
<td>February 23/99</td>
<td>March 23/99</td>
</tr>
<tr>
<td>Final Papers</td>
<td>June 1/99</td>
</tr>
</tbody>
</table>

**2nd Large Installation System Administration of Windows NT Conference (LISA-NT)**

<table>
<thead>
<tr>
<th>DEADLINES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper Submissions</td>
<td>Notification to Authors</td>
</tr>
<tr>
<td>Final Papers</td>
<td>Final Papers</td>
</tr>
<tr>
<td>February 23/99</td>
<td>March 23/99</td>
</tr>
<tr>
<td>Final Papers</td>
<td>June 1/99</td>
</tr>
</tbody>
</table>

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

2 IN THIS ISSUE . . .

IMHO
3 Pulling on One End of the Rope
   by Jordan K. Hubbard

LETTER TO THE EDITOR
5 Clarification
   from Jim Reid

CONFERENCE REPORTS
6 USENIX 1998 Annual Technical Conference

SAGE NEWS AND FEATURES
33 Home Sweet Home
   by Tina Darmohray
34 Flattening the Curve
   by Hal Miller
35 Taming the Certification Beast
   by David Conran, Barb Dijker, Tim Gassaway, and Kim Trudel
37 Speakers Bureau
   by Tim Gassaway and Helen Harrison
38 Please Quit
   by Tom Limoncelli
39 Effective Perl Programming:
   Simplicity Is a Good Object
   by Joseph N. Hall

FEATURES
43 Interview with L. Peter Deutsch
   by Stig HackVæn
54 Source Code UNIX on the PC
   by Bob Gray
60 The Security Model of JDK 1.2
   by Dario Forte
62 Java Performance
   by Glen McCluskey
64 Using Java
   by Prithvi Rao
68 Musings
   by Rik Farrow

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

BOOK REVIEWS
75 The Bookworm
   by Peter H. Salus
77 Java Network Security
   Reviewed by Terry Rocker
78 Sun Performance and Tuning:
   Java and the Internet
   Reviewed by Andrew Hume
79 LDAP: Programming Directory-Enabled
   Applications with Lightweight Directory
   Access Protocol
   Reviewed by George W. Leach
80 Managing Mailing Lists
   Reviewed by Rick Umaili

USENIX NEWS
81 Letter from the President: Rainless in Seattle
   by Andrew Hume
82 Summary of USENIX Board Meetings
   by Ellie Young
84 Complete Set of CSRG BSD Releases Available
   by Kirk McKusick
84 Twenty Years Ago in ;login:
   by Peter H. Salus
85 USENIX Annual Awards
85 Tom Money
85 USACO Camp Completed
86 USACO – Letters from Students

ANNOUNCEMENTS AND CALLS
88 LISA ’98
91 2nd IEEE Workshop on Mobile Computing Systems and Applications
   (WMCSA ’99)
92 1st USENIX Workshop on Intrusion Detection and Network Monitoring
94 2nd USENIX Symposium on Internet Technologies and Systems (USITS ’99)
96 motd
   by Rob Kolstad

October 1998 ;login:
The secret is out. A seven-page cover story in *Forbes's* August 10 issue, complete with arty photos of the likes of Linus Torvalds, Richard Stallman, and Larry Wall, has told the business community about the freely redistributable software phenomenon. Fame (along with, in some cases, fortune) is coming to developers who believe the best way to produce good software is to do it openly and cooperatively. USENIX, of course, has long been in the know, and luminaries and aficionados of the freeware world have been gracing our conferences for years.

This issue of *login* (not to be outdone by some capitalist rag) has an especially strong freeware focus. Start with the heartfelt admonishments in Jordan Hubbard's opinion piece on the opposite page about the dark side of the world of free UNIX. From there you can check out summaries of several sessions of the FREENIX track (along with the refereed-paper and invited-talk tracks) at the 1998 USENIX Annual Technical Conference in New Orleans, as well as Peter Salus's "Free Stuff" opinion piece. In the Features section, Bob Gray's column on source code UNIX for the PC showcases a myriad of freely available applications, and discusses how to obtain, install, and maintain them. And we are delighted to present a fascinating interview with L. Peter Deutsch, the developer of Ghostscript.

You'll also find some in-depth book reviews, a bunch of Java articles, more on object-oriented programming with Perl, SAGE and USENIX news galore, and Rik Farrow's musings on network security. Oh, yes ... and some rather unique photographs suitable to the season.
imho: pulling on one end of the rope

by Jordan K. Hubbard

Jordan was one of the original founders of the FreeBSD Project and is its public relations officer and release engineer, as well as President and CEO of FreeBSD, Inc. He will chair the FREEUNIX track at USENIX '99.

<jkh@time.com>

As I write this article, FreeBSD has just had its fifth birthday. As one of the original founders, I have also been around to watch it grow from what was essentially a three-man operation with a couple of dozen developers to a project with several hundred developers and well over a million users worldwide. It is now used as a server OS by large companies, from Yahoo to Oracle, and by casual users as a desktop workstation solution. Sales of FreeBSD installation media and FreeBSD-based products now run into the tens of millions of dollars a year. Not a bad achievement for free software!

At the same time, to give everyone their just due, I’ve watched Linux grow from a personal project by a Finnish computer science student into an international phenomenon, many of its users embracing it with the same degree of fervor usually reserved for rock stars and single-malt Scotch whiskey. Many former Windows and Macintosh users who probably would not have even mentioned UNIX without a grimace before are now preaching the gospel of Linux with the fire of the newly converted, and that’s a good thing. Linux has, in no small part, helped to bring about what amounts to a second renaissance for UNIX, synergy with the Internet boom propelling it far ahead of any previous efforts of this type. My personal OS preferences do not prevent me from admiring the sheer scale of what Linux has accomplished, believe me!

So then all is rosy and wonderful, and this is basically another feel-good editorial about the magic of free software, right? Wrong, unfortunately. There are snakes in the garden of UNIX, and, surprise surprise, they’re the very same snakes we’ve seen throughout its somewhat checkered history. The snakes of Unrestrained Ego and Not Invented Here still slither freely, it mattering not whether the ground they slither over is free or commercial, and (as usual) they’re causing us some serious problems.

To illustrate just how serious those problems can get, let’s go back about eight years to a period known as the “GUI wars”—a time of all-out battle between OpenLook, Motif, and a host of lesser contenders. For those who were “there,” the event probably needs no further description—the very mention of the GUI wars is enough to send cold shivers down the spine of even the most hardened UNIX devotee. Not only did these wars cost UNIX the desktop; they cost it a significant number of the few remaining ISVs (independent software vendors) that it had. I was working for Lotus at the time, and I can tell you that the idea of supporting two different standard UIs impressed no one (and if you wanted to run on Suns and some other UNIX platform back then, that’s what you had to do). A number of smaller ISV partners of ours decided then and there that they’d simply back the Windows and Macintosh platforms and leave the UNIX folks to claw and scratch at one another in peace.

The GUI wars also only added to an already tenable situation brought on by the SYSV/UCB split (AIX/HP-UX vs. Ultrix/SunOS, etc.), another ugly chapter that I won’t go into here, making the typical ISV’s life a real poring hell in return for access to a market that was already dwindling in comparison to the booming Windows market. If it were at all possible for the UNIX market to sabotage itself more effectively during that period, I’m not sure how it could have been accomplished. It was one big, massive Jonestown-style poisoned Kool-Aid party and probably represented one of the few activities the UNIX community managed to unite itself successfully behind. When unification of the GUI finally did occur, backing a standard (Motif) which remained commercial and out of reach of the casual user, it was too little too late and, many would also say, entirely the wrong choice.

“That all sounds pretty pathetic. How and why did this happen?” I hear you ask. Well, first there was the matter of Ego. Everyone wanted to be the one to “own” the standard for how the GUI on this nifty new X windows system would look. The dominant corporate players that eventually emerged also decided that they wanted to either sell their technology or keep it proprietary and distribute it only with their own workstations. Both concepts, of course, completely missed the point and caused incalculable damage in so doing.

The X windows system itself was popular principally because it was a free and open standard with a nonaggressive copyright. (You could productize it if you wished without being compelled to give the source away, helping to get it past the usual corporate legal department blockades without a fuss.) And it was highly portable—I remember being very chuffed that I could now use the same window system on the IBM PC/RT, Sun 3/50, and DEC MicroVAX machines in my office, a real luxury at the time. The people who wanted to create follow-on standards that they alone controlled were simply not positioning themselves to take advantage of the X user community’s rapid growth, nor could they leverage the work of all those potential volunteer programmers while the sources were being kept under lock and key, so to speak.
NIH was also a big factor, nobody liking anyone else's standard and deciding that a perfect standard was far preferable to having any standard (de facto or not) at all. Of course, no such degree of perfection was ever achieved, and the greater good was sacrificed in pursuit of short-term gains that never even really materialized. The opportunity for an effective and open standard was not only thrown away; it was thrown away for nothing. Thus ended the first age of UNIX, many (including myself) believing that this was quite possibly The End because its community of developers, although highly capable tacticians, had proved to be abominable strategists and seemed to be winning battles in order to lose wars with depressing frequency.

Wind the clock forward to today, almost a decade later, and we find many of those things happening again and for exactly the same reasons. Don't get me wrong—it is a perfectly natural human desire to form clans and proudly wear the clan colors on Robert the Bruce Day, or whatever. Diversity and competition are good things that encourage innovation and inspire people to greater heights of productivity. When you add Rampaging Egos to the mix, however, things get messy very quickly as the various clans decide that they want to be the clan, the only ones in line when the awards for best clan colors or most unique sporran are being handed out. Before long, rival clans are firing flaming arrows at one another and launching raids into others' camps, generally making life difficult all around for a lot of folks who would really rather just get on with the business of living.

So it is today in the world of free software. Even though any marginally sane person would be appalled at the sight of two organizations like C.A.R.E. and the International Red Cross fighting one another for the privilege of feeding starving children in Africa, for some reason, the same behavior seems okay if it's just a bunch of people who write free software doing it. I don't mean to equate the process of feeding starving children with that of writing free software, far from it, but they're both "benevolent activities" that one would certainly hope could transcend any rivalries in carrying out their good works.

NIH is also still alive and well, many people choosing to do the same work over (and over) again just because it wasn't someone from their clan who wrote it or they have some deep-seated prejudice against anything done by clans who put green before red in their kilts. It's simply ludicrous a lot of the time! More importantly, the larger picture is being lost again, just as the entire UNIX world is getting a second chance at life (a privilege not usually afforded to software of this nature—once it dies, it generally stays dead). The larger picture that people are losing sight of is that we're all truly in this together. Even if we don't explicitly go out of our way to help one another, at the very least, we shouldn't be doing our damnedest to kick the crusts out of under one another.

A good example of keeping sight of the larger picture is FreeBSD's attitude toward its UNIX emulation. It's not only very important to us that FreeBSD continue to run Linux binaries effectively; it's also what we suggest to those ISVs who are coming back somewhat cautiously to this "new" UNIX market and obviously want to maximize their gains while minimizing risk. We tell them to port to Linux and not FreeBSD, even though we'd certainly love to have native binaries for anything and everything. By telling them to port to Linux first (or at all), we are giving them the best advice on how to get access to the widest possible segment of the free software market, one that includes, but is not limited to, us. That is the kind of "what is best for all the clans?" thinking I actively try to promote and essentially why I am taking the time out to write this editorial.

After five years of intermittent warfare, not just between the Linux/BSD camps, but also within the various UNIX and *BSD camps themselves (serving only to prove that any clan can and will fight another, even when they're all related), it's also not going to be one giant hug fest from now on just because people like me stand up and say that everyone really ought to get along. Life's not that simple. What we can do, however, is to continue to strongly promote any and all ties between the various free software groups and also actively encourage users to familiarize themselves with each and every one of the various types of free software out there, whether they're currently "pledged" to a given cause or not. Not only will this experience help to shatter some of the walls of mistrust and general acrimony between the various clans, but it can also benefit those who are firmly convinced that they wish to stick with a certain one.

Say, for example, that someone fairly prominent in the Linux community popped up and told various users that they might want to give FreeBSD a whirl as well sometime, just to check out what it has to offer lately. Those users would then likely as not come back with a set of things they liked and didn't like about the experience. The things they liked could serve as rich idea fodder for anyone in the Linux camp wishing to do similar things. Believe me, every time a new release of Linux comes out, I do install it on a test box here and check it out. I'm not proud (or stupid) and am more than happy to adopt any feature or concept that looks like a good idea if I've got the time to do it.

Finally, various organizations are also recognizing the danger signs again and, thankfully, beginning to do something
about them. USENIX, long a standard bearer for all UNIX-related evangelical activities, launched the FREENIX track at this year's USENIX Annual Technical Conference in New Orleans, a special track devoted to free software of all types. The over 1,400 attendees of this conference raved FREENIX as one of the top attractions for them, and next year's conference (in Monterey, CA) will feature an even more ambitious FREENIX track (if I have anything to say about it, anyway).

The Call For Papers is out for this event <http://www.usenix.org/events/usenix99> and all of you in the free software world, be you aligned with Linux, GNU, FreeBSD, OpenBSD, NetBSD, Samba, or whomever, are more than encouraged to present a paper at this conference or simply show up as attendees. If you want more information on this, simply send me email, and I'll be happy to send you a description of the event (which I'm also chairing next year). Events like this provide a very valuable once-a-year opportunity to learn that the other free software folks don't bite (much) and catch up on the details of what they're up to. More important, meeting face to face is almost always a much better way of building bridges because potentially sensitive topics can be discussed without someone going ballistic at a misparsed phrase or an attempted joke that fell flat. You'd be amazed at how conflicts that have burned for months can be suddenly and easily resolved with one short 30-minute talk over a cup of coffee.

Above all, whether you can attend events like FREENIX or not (and heck, if it's too far away, why not create your own?), please do your very best to keep an open mind when it comes to all the free software out there today and remember that, like the Red Cross, we're all in this together in the service of a very good cause. The last thing we need is dissension and discord to tear us apart and, ultimately, (again) lead to our stealing defeat from the jaws of victory.

**Letter to the Editor**

**Clarification**

There are two subtle but significant errors in the report of my talk at SANS98 in the August issue of *login* that I feel should be corrected:

1. Origin b.v. is a global IT service company which is part of the Philips group. One of the services provided by one of the divisions of Origin is the management and operation of the Philips Intranet. It is definitely not the case that "the Origin group" and the Philips Intranet are one and the same.

2. We specifically chose BSD/OS as the operating system for the backbone nameservers, not just "a BSD-based Unix." The main reason for using BSD/OS was that this is the reference and development platform for BIND. The other important consideration was that BSDI could provide an appropriate level of support for something as vital as the DNS platform in a huge, global Intranet. These criteria did not apply to other BSD-based UNIX systems, most of which can run on other CPU architectures such as SPARC.

It's possible that my presentation at SANS98 may not have made these points clearly enough, so I would like to take this opportunity to set the record straight.

Jim Reid
Origin TIS-INS
This issue’s reports focus on the USENIX 1998 Annual Technical Conference, held in New Orleans, Louisiana, on June 15-19, 1998.

Our thanks to the Summarizers:

Peter Collinson
<pcc@hillside.co.uk>

Dave Essex
<edsall@astate.edu>

Kevin Fu
<fu@bub@mit.edu>

Steve Hanson
<shanson@htc.honeywell.com>

Jon Howell
<jonh@cs.dartmouth.edu>

Tom M. Kroeger
<tmk@cs.ucsc.edu>

Brian Kurotsuchi
<bkurots@hornet.csc.calpoly.edu>

Branson Matheson
<branson.matheson@ferginc.com>

Keith Parks
<kpark@uhura.cc.rochester.edu>

Jason Peel
<jsym@acm.org>

Terri Watson Rashid
<terri@msit.net>

Karen Reid
<kare@acg.toronto.edu>

Peter H. Salus
<peter@pedant.com>

Gus Shaffer
<shaffer@monkey.org>

Jim Simpson
<jsimpson@monkey.org>

Vikas Sinha
<vikas@athavan.eng.wayne.edu>

Our thanks to the Photographer:

Sean Cox
<scux@factset.com>

---

USENIX 1998 Annual Technical Conference

NEW ORLEANS, LOUISIANA
June 15-19, 1998

KEYNOTE ADDRESS

Science and the Chimera
James “The Amazing” Randi
Summary by Peter Collinson

James Randi’s keynote talk kicked the 1998 Technical Conference into life. His talk zipped by, and we walked out at the end having seen several examples of his work, his magic, and his mission to expose the quacks and charlatans who often employ technology to fool us, the gullible public.

James’s roving, debunking eye has been aimed in many directions, from harmless mentalists to the somewhat more serious faith healers in both the US and the Philippines whose activities not only net them huge sums of money, but also give false hope to people whom medical care cannot cure. He looked at homeopathic cures which are now sold in many drugstores. These cures are simply water because the original chemical has been diluted $10^{2500}$ times.

His talk ended on some serious notes:

- We need to teach our children to think critically so they stop being fooled
- We need to stand up and expose fraudulent use of science and technology.

His Web site is <http://www.randi.org> and is well worth a visit.

REFEREED TRACK

Session: Performance 1
Summary by Tom M. Kroeger

Scalable Kernel Performance for Internet Servers Under Realistic Loads
Gaurav Banga, Rice University; Jeffrey C. Mogul, Digital Equipment Corp., Western Research Lab

This work, presented by Gaurav Banga, earned both the Best Paper and Best Student Paper awards at the conference. It examined an inconsistency between the observed performance of event-driven servers under standard benchmarks, like SPECWeb96, and real workloads. Banga and Mogul observed that in a WAN environment, which is characterized by inherent delays, a server is forced to manage a large number of connections simultaneously. Because commonly used benchmarks lack slow client connections, they failed to test system behavior under such conditions. From this observation they developed a new WAN benchmark that tried to model slow connections.

They then profiled the Squid proxy server under both a standard benchmark and their new benchmark. The standard benchmark showed no specific procedure in the system to be a bottleneck, but in the WAN benchmark the kernel procedures for select and file descriptor allocation (ufalloc) accounted for 40% of the CPU time. With this information, Banga explained how they examined the implementations of select and ufalloc.
The `select` system call in Digital UNIX (and in fact in most UNIX variants) was designed at a time when several hundred connections as an input argument would have seemed extreme. Banga explained how the current implementations scale poorly because of linear searches, layered demultiplexing, the linear system call interface, and the pressure that these functions put on the CPU data cache. The key insight here is that `select` wastes a significant portion of time rediscovering information that previous stages of protocol processing had available. Using hints to transfer this state information, they were able to prune the scans that `select` needed to perform.

Next Banga explained how they examined `ufaalloc`. This procedure is called every time a new file descriptor is allocated. Again a linear search was at the heart of the problem. UNIX semantics state that `ufaalloc` must provide the lowest free descriptor; this prevents the use of a free-list. To solve this problem, the authors reimplemented `ufaalloc`, adding a two-level tree of bitmaps to indicate available descriptors. This new implementation changed `ufaalloc`'s complexity from \(O(n)\) to \(O(\log n)\). It also provided for better cache behavior requiring two memory reads vis-à-vis a sequential scan that would trash the entire data cache. Lastly, it provided better locking behavior because of a shorter critical section.

Banga explained how they set up two testbeds to evaluate the effect of these changes. First, using the WAN benchmark on both the Squid proxy and the thttpd Web server, they showed that scalability with respect to connection rate and connection count was significantly improved. They then tested their changes under a live load, i.e., on Digital's Palo Alto Web proxies. Again, they found significant improvements in performance from the modified systems.

**Tribeca: A System for Managing Large Databases of Network Traffic**

Mark Sullivan, Juno Online Services; Andrew Heybey, Niksun, Inc.

Mark Sullivan presented the Tribeca system for network traffic analysis, which the authors developed after noting how the use of ad hoc analysis programs resulted in redundant efforts. They have developed a general query system for an environment where network traffic data streams by at rates of up to 155 megabytes per second. They observed that a typical relational database system would not be effective for network analysis for the following reasons. Both the data and storage media are stream-oriented. Relational database systems do not normally handle tape data well, and tape data are commonly used for network traffic analysis. The operators needed are more like those in temporal and sequential databases. Traffic analysis commonly requires running several queries during a single pass. Lastly, relational database systems rarely consider the memory capacity of the system on which they are running. The Tribeca system addresses all of these issues, but also differs from conventional relational databases in that it does not support random access to data, transactional updates, conventional indices, or traditional joins. Tribeca takes its source data from either a live network adapter or tape data.

The query language in Tribeca is based on a data description language. The different protocols are expressed as different data types; this language then allows the user to create new types by extending compiled-in types. This language also provides support for inheritance, arbitrary offsets, and bit fields. Each query has one source stream and one or more result streams. To manipulate these streams, Tribeca provides three basic operators: qualification, projection, and aggregation. Additionally, the query language provides for stream demultiplexing and remultiplexing. Finally, the language also provides a method for operating on windows over the stream.

Tribeca's implementation shares several similarities with traditional relational database systems. Queries are compiled into directed acyclic graphs. These graphs are then optimized to improve performance. The basic data management for Tribeca makes use of existing operating system support for sequential I/O, and because data are not updated, no support for concurrency control is needed. Special attention was paid in the implementation to minimize internal data copying. Additionally, the optimizer also works to ensure that a query's intermediate state can be fit into the memory available.

The authors presented some basic tests to examine the overhead incurred. They compared the overhead for a basic query with that of the standard UNIX program `dd`: `dd` used 68% of the CPU on a Sparc 10, but Tribeca used only 70%–75% of the CPU time. Lastly, they compared the performance of Tribeca to that of programs written directly to execute a specific query. The results showed that Tribeca incurs between 1% and 7% overhead. The authors concluded by noting that the increased flexibility and convenience provided by Tribeca are well worth the minimal overhead introduced.

**Transparent Result Caching**

Amin Vahdat, University of California, Berkeley; Thomas Anderson, University of Washington

Amin Vahdat presented a system (TREC) developed by the authors to track output of a process's execution based on the inputs. Using this information, TREC provides a framework to make use of previously existing outputs and observe process lineage and file dependencies. Implemented through the use of the `proc` file system under Solaris, TREC intercepts the open, fork, fork1, creat, unlink, exec, execve, rename, and exit...
system calls. By catching these calls TREC is able to record a process’s input files, child processes, environment variables, command line parameters, and output files.

After explaining the basic architecture, this work addresses the limitations of TREC. In order to address concerns about the performance overhead from intercepting system calls, the authors examined the added overhead for a test program that simply called open and close in a loop and two typical application executions. The test program saw 54% overhead but the typical applications saw only 13% and 3%. The authors observed that the overhead is directly proportional to the system call rate and noted that a kernel-level implementation would significantly reduce this overhead.

The authors also noted several requirements for TREC to produce correct results. The program itself must be deterministic and repeatable, it cannot rely on user input, and interaction with environment variables must be reproducible. File contents must be static, files such as /dev/rtc0 could produce incorrect results. File contents must be changed locally; for example, NFS-mounted files could be modified on a remote machine and not be reported to TREC. Processes base their results on communication with remote servers cannot, in general, be correctly tracked. Lastly, a program must complete successfully.

After detailing the limitations of this system, the authors provided three examples of applications that use the TREC framework: unmake, transparent make and a Web cache that enables server and proxy caching of dynamically generated Web content.

Unmake provides a facility for users to query the TREC framework to determine how specific output files were created as well as questions about process lineage. Transparent make provides an alternative to make that automatically determines file dependencies. Instead of providing a possibly complicated Makefile, the user provides a simple shell script that performs the complete build sequence. Once transparent make has observed this shell script and each program’s input and resulting outputs, transparent make can be used for subsequent builds to execute only those commands for which the inputs have changed in some manner.

This system has the following advantages: user errors in dependency specification are avoided, dependencies are updated as the input is changed (e.g., a header file is added to a program being compiled), and the users are saved from learning the Makefile specification language. Transparent make provides two current variants: a passive version that will update output files when executed and an active version that registers a callback with the TREC framework. For this active version, upon observing changes to specific input files if a callback was registered for that file, then transparent make will prompt the user for reexecution of the registered modules.

The third example of the uses for the TREC framework is a modification to the Apache Web server to cache the results of cgi scripts. The authors modified an Apache server to store copies of the results from a cgi program’s execution indexed by the program’s parameters. When the cgi program is called, the server first checks for a pregenerated result for the requested program and parameters. If these exist, it responds with the contents of that file instead of executing the cgi script. To invalidate these dynamic cache entries, the TREC framework is then used to profile the execution of each cgi program. When an input to this program is observed to change, TREC notices a registered callback similar to those for the active version of transparent make. This callback invalidates the cached result. Comparing the two servers (caching versus forking) with a basic cgi script, the authors observed a 39% improvement in average response time.

Session: Extensibility
Summary by Karen Reid
In her introduction to this session, session chair Terri Watson Rashid noted that the three papers represent a wide range of how extensibility can be used. The first paper discusses a way to extend operating systems; the second describes an extension to the SPIN operating system; and the final paper presents methods of extending applications.

SLIC: An Extensibility System for Commodity Operating Systems
Douglas P. Ghormley, University of California, Berkeley; David Petrou, Carnegie-Mellon University; Steven H. Rodrigues, Network Appliance, Inc.; Thomas E. Anderson, University of Washington

The extension mechanism described by David Petrou makes it possible to add functionality to commodity operating systems with no changes to the operating system. These extensions allow system administrators to easily add trusted code to the kernel to fix security flaws, take advantage of the latest research, or better support demanding applications.

SLIC is built using the technique of interception: capturing system events and passing them to the extensions. The system has two components: the dispatchers that catch events, call the extensions, and provide the API for the extension framework and the extensions themselves that implement new functionality.

Petrou described three examples of extensions implemented using SLIC. The first fixes a security flaw in the Solaris admintool. The second extension adds encryption to the NFS file system. The third one implements a restricted execution environment by filtering system calls.

Petrou and his co-author/slide-turner, Steven Rodrigues, pulled out an extra slide to answer the first question of how to manage the order that the interposi-
tions were applied. Unfortunately, the answer was that although the syntax for composing interpositions is not a problem, determining that a series of interpositions is semantically correct is a difficult, and as yet unsolved, problem.

The second question confirmed that interpositions can be applied only to interfaces at the kernel boundary. Petrour noted that SLIC could not be used to change the paging algorithm for an application.

When asked whether the extensions would primarily be useful to prototype kernel extensions or for low-volume extensions, Petrour claimed that the examples showed that extensions could be used to solve a wide range of problems, not just those in a research environment.

David Korn asked if interpositions could store data on a per process basis. Petrour replied that the extension can store per thread state.

The remaining questions concerned the portability of the interposition system. Petrour argued that the extensions should be quite portable, but that the dispatchers needed to be ported to other architectures. They are currently working on porting the dispatchers to Linux.

A Transactional Memory Service in an Extensible Operating System
Yasushi Saito and Brian Bershad, University of Washington

Yasushi Saito presented Rhino, an extension for the SPIN operating system that implements a transactional memory service. A transactional memory service uses memory-mapped files and loads and stores to implement transactions that are atomic, isolated, and durable (ACID). Transactional memory can be used to support object-oriented databases and persistent low-level data structures such as filesystem metadata.

Saito contrasted his work with user-level implementations of transactional memory by highlighting several problems with the user-level approach. First, context switches for the signal handler and protect() incur too much overhead. Also, the user-level implementation requires fast interprocess communication. Finally, buffering problems occur because the user-level process that is mapping database files into memory has no control over the paging. Double buffering occurs when the memory system decides to reclaim pages and swaps out database pages instead of writing them back to the database file.

The approach taken by the authors to solve these problems is to do everything in the kernel. The SPIN extension gives them fast access detection through the kernel page fault handler and efficient memory-mapped buffers through cooperation with the kernel memory manager.

Three options for buffer management were discussed. The first relies on the user to notify the extension (by calling trans_setrange()) about a region that will be modified. This method is efficient for small transactions, but doesn’t scale well when the number of setrange() calls is high. The second option is to log the entire page when at least one byte is modified. This approach works well for large transactions, but is costly for small transactions. The third method computes and logs the diffs between the page and the updates. Page diffing combines the advantages of the previous two approaches, but incurs significant overhead.

Saito compared the performance of the SPIN-based transactional memory service to one implemented at user-level on UNIX and to ObjectStore, a database management system. The SPIN-based system consistently outperformed the other two for the given workloads.

Terri Watson Rashid asked Saito to comment on his experiences implementing kernel extensions on SPIN. Saito was reluctant to make any strong comparisons between implementing the user-level UNIX implementation of Rhino and the SPIN extension, but said that debugging facilities for SPIN extensions made writing kernel-level code much easier.

Dynamic C++ Classes
Gisli Hjálmtýsson, AT&T Labs-Research; Robert Gray, Dartmouth College

This work, presented by Gisli Hjálmtýsson, was motivated by the desire to allow “hot” updates of running software. In other words, they wanted a system that allows users to insert or replace components of a software system while it continues to run. This technique can be applied to domains such as network processing, where it is often highly undesirable to halt and restart programs.

The authors achieved their goal of updating running code by implementing a library to support dynamic C++ classes. This approach was chosen because C++ is widely used, high performance can be maintained, and program abstractions can be preserved. Dynamic classes allow for runtime updates at the class granularity. New versions of existing classes can be installed, and new classes can be added. However, they require that class interfaces be immutable.

One big question is how to dynamically link in new or updated classes. Hjálmtýsson proposes three different approaches to updating objects: imposing a barrier, where no new objects may be created until all objects of an older version have expired; migrating old objects to their new version; and allowing multiple concurrent versions of each class. The disadvantage of the barrier approach is that it is equivalent in some ways to halting the program and restarting. The migration approach is hard to automate efficiently, so the authors chose the third
approach of allowing concurrent versions of a class.

Dynamic classes have two parts: an abstract interface class and an implementation class. An interface monitor, implemented as a class proxy, screens messages that pass through dynamic class interfaces and direct them to the correct version of the class. Two levels of indirection are used: one to map to the implementation and the other to map the methods within a version. This approach requires that all public methods of a dynamic class be virtual.

Using the factory pattern, an object of a dynamic class is constructed by calling the default constructor, which locates and loads the dynamic shared library, calls the constructor for the correct version, and stores a pointer to the version of the implementation class in the proxy.

Three different templates for proxies are defined. They differ in ease of use, functionality, and performance. The high-functionality version of the template allows multiple implementations of a dynamic class interface as well as multiple active versions of an implementation. The medium-functionality version allows multiple versions, but not multiple implementations of a dynamic class. Both the medium- and high-functionality versions implement methods to invalidate other dynamic class versions. Finally, the low-functionality, but highest performance, version of the template allows multiple concurrent versions of a dynamic class, but old versions cannot be invalidated.

The flexibility of dynamic classes does not come without a cost. Each instance requires space for three or four extra pointers. The method invocation overhead is approximately doubled because of the extra checks required and because some of these checks cannot be optimized (by most compilers) because a method that can throw an exception cannot be inlined.

Hjálmtysson used mobile agents as an example of how dynamic classes could be used. Different versions of the agents for different architectures can be downloaded so that agents can be instantiated on a variety of platforms.

Hjálmtysson concluded by describing dynamic classes as a lightweight mechanism to update running code that preserves type safety and class abstractions. It compiles on SGI, Sun, and Windows 95/NT and is available from AT&T by contacting the authors.

During the question period, the similarity between dynamic classes and Corba or ActiveX was noted. Hjálmtysson acknowledged the similarity and claimed that dynamic classes have less overhead around them and are a more lightweight approach. Not surprisingly, other questions focused on how a similar system might be written for Java, whether the Java Virtual Machine (JVM) would have to be modified, and how the JVM might be forced to unload classes. Hjálmtysson said he believed that it may be possible to implement dynamic classes without modifying the JVM, but that the class loader would need to be modified, making it less portable.

**Session: Commercial Applications**
Summary by Brian Kurotsuchi

Each of the papers in this session dealt with the low-level, behind-the-scenes operating systems internals, specifically, the filesystem and virtual memory subsystems.

**Fast Consistency Checking for the Solaris File System**
J. Kent Peacock, Ashvin Kamaraju, Sanjay Agrawal, Sun Microsystems Computer Company

Kent Peacock presented his group’s work with the optimization of the native Solaris UFS filesystem to improve performance while supporting the semantics of NFS services. He explained that NFS semantics require data to be committed to stable secondary storage before the NFS transaction can be completed. This requirement unfortunately precludes the use of filesystem block caches, which are generally used to improve read/write performance. In order to overcome the synchronous write requirement, they decided to use some type of fast NVRAM storage medium to provide safe buffering of the physical storage device; they first used a UPS on the system, then actual NVRAM boards. With this NVRAM solution, they gained performance by not having to wait for slow secondary storage to complete before acknowledging the NFS transaction. Peacock also mentioned that they tried traditional logging (journaling) to the NVRAM, but were unable to meet performance requirements using that approach.

The second issue that Peacock addressed was that of filesystem performance at runtime and when `fsck` is required to check the filesystem. In order to do this, they added additional data structures to the on-disk filesystem representation and modified some of the ways in which metadata are juggled. The areas Peacock focused on were busy bitmap methods and their changes in the use of indirect blocks.

The Solaris UFS filesystem is divided into cylinder groups, each of which contains a bitmap of free blocks. An `fsck` involves checking this data in each cylinder group on the disk, an operation that can take some time. In order to reduce the number of metadata structures that need to be checked during an `fsck` run, there are special bitmaps in parallel (physically) with the free block bitmap. These new bitmaps indicate which blocks and i-nodes in that cylinder group are being modified (are busy). Each cylinder group can then be flushed and marked as "stable" asynchronously by a kernel thread. This can greatly reduce the time needed to do an `fsck` because only cylinder groups that are still marked "busy" need to be checked.
An interesting variation that Peacock's group came up with is the handling of indirect block maps to reduce the number of writes to disk. Indirect blocks are normally stored separately from the i-node, hence in a different block on the disk. Updating a large file that requires the use of the indirect blocks incurs a read and write of at least two blocks instead of one (i-node only vs. i-node + indirect block[s]). To defer the need to deal with the additional blocks, temporary indirect block storage is interleaved on odd i-nodes in the i-node table. Each time an indirect block is needed, it is written into the i-node slot adjacent to the file's i-node, requiring only a single write operation. When the adjacent i-node storing the indirect pointers is full, it is flushed to the traditional indirect block (hence deferring all indirect block I/O operations until this time).

In conclusion, Peacock reminded us that NFS is inherently disk bound because of the synchronous write requirements. His group was able to overcome this by using NVRAM storage to satisfy NFS semantics and attain high throughputs performance. On top of that, they were able to make additional gains by modifying UFS to use the indirect block cache and busy maps. The data gathered by Peacock's group seem to indicate runtime and cache performance above and beyond that of standard UFS and the widely used Veritas File System. This modified filesystem is in use on Sun's Netra NFS server products and may appear in a future Solaris release.

The audience questions indicated some skepticism on the Veritas benchmarks that were stated. An important question concerned NFS version 2 versus version 3, for which Peacock said they found a smaller performance gap between their Netra NFS implementation and NFS version 3.

**General Purpose Operating System Support for Multiple Page Sizes**


Narayanan Ganapathy gave an excellent presentation that outlined the advantages of using virtual memory page sizes above the normal 4k and a walk-through of how they implemented this idea in IRIX (v6.4 & 6.5). There are some applications out there that could see improvements in performance if they could use large pages of memory versus small (4k) pages. Much of the overhead for an application that deals with large sets of data could be TLB misses. Ganapathy presented an explanation of the reason behind and experience they had at SGI while retro-fitting the IRIX virtual memory system to allow processes to use multiple page sizes.

One of the goals in designing this multi-sized paging system was to minimize the change to existing operating system code and maintain the flexibility and compatibility with existing application binaries. The implementation they chose makes changes at a high level in the virtual memory subsystem, the per process page table's entries (PTEs). This is the map of all pieces of memory that can be accessed by a process. To support the large pages, each PTE has a field that states the size of that page (4k-16M on the R10000). The memory area to which that page refers may be handled at a lower level by a pfdat (page frame data) structure, which they chose to keep as statically sized 4k pieces for compatibility. A major advantage to doing things this way is that multiple processes can still share memory, but the size of the area that each of them sees in its page table does not have to be the same. One process can map 16k pages while another maps 4k pages, both of them ultimately referring to the same 4k pfdat structures (in effect the same physical memory).

Allowing processes to manipulate their PTEs in this way produced some interesting problems, such as memory fragmentation, fast processing of TLB misses, additional systems calls and tools to manipulate the page sizes. Fragmentation is avoided by intelligent allocation of pages through the use of maps for free segments of memory of similar size and a "coalescing daemon" to defragment memory in the background using page migration to rearrange. To prevent all processes from going through extra code even when they are using the default page size, IRIX provides the ability to assign a TLB miss handler on a per process basis. A system call has been provided to change the page sizes, plus tools to allow normal binaries to be configured to use large pages without recompilation.

In closing, Ganapathy mentioned the possibility of intelligent kernels that could automatically choose page sizes for a process based upon TLB misses.

**Implementation of Multiple Pagesize Support in HP-UX**

Indira Subramanian, Cliff Mather, Kurt Peterson, and Balakrishna Raghu Nath, Hewlett-Packard Company

The final presentation in this session was given by Indira Subramanian. Although this presentation was on the same subject as the previous one by the SGI group, they were well coordinated and did not seem like redundant subject matter.

As in the Silicon Graphics implementation, the HP group wanted to minimize kernel VM subsystem changes. Their implementation also avoids changes to the low-level VM data structures and implements variable sized pages at the PTE level.

Allocation and fragmentation management is governed by an implementation of the buddy system, with the pools representing free memory regions from 4k to 64M in size. The page management subsystem uses two different strategies to deal with requests for new pages. The VM
system will automatically combine pages to create larger pages as soon as a page is freed. Next, pages not currently in the cache will be evicted and coalesced into the free pool. The last resort is to evict and coalesce pages currently in the cache. Using this algorithm should give the greatest chance for pages to be retrieved out of the cache.

A single modified HP-UX page fault handler is used for all page faults that occur in the system. It is capable of dealing with copy-on-write, copy-on-reference, zero filling, and retrieval of large pages when necessary. It is possible to provide the page fault handler with a page size hint either through the use of a utility program (chatz) or through an intelligent memory region allocation routine. This "hint" allows the page fault handler to bypass page size calculation and allocation if it can determine that the default of 4k is going to be used. The basic idea was that if the page size hint shows that the new page will be probably going to be greater than 4k, the fault handler would take the following steps: (1) calculate the size, (2) allocate the necessary region, (3) add the necessary translations, and (4) zero fill the page if needed. The size calculation and large region allocation can be completely skipped if the new page will be a simple 4k, hence preserving performance in those cases.

Perhaps the most gratifying part of this presentation was the place where Subramanian spent a lot of the time – the graphs. By experimenting with applications that use memory in different ways, the data showed that one large page size was not suitable for all situations. In one application, there was a very high TLB miss rate while using 4k pages, but a much better hit rate with 4M pages. As you would expect, the law of diminishing returns kicked in when excessive page sizes were selected.

In conclusion, Subramanian reminded us that the page sizes are not promoted (combined to make larger pages) except when the page fault handler identifies regions experiencing large TLB miss rates. Reducing TLB misses was the project goal, which they accomplished through adding the ability to use large pages and by having the VM system dynamically monitor memory usage and adjusting page sizes to reduce TLB misses at runtime.

One unfortunate factor in this presentation was lack of time. Our presenter was fielding some very interesting questions from the audience and flipping at blinding speed between graphs of data right up to the end.

**Session: Performance II**

Summary by Vikas Sinha

This was the second of system performance-related sessions. Papers focusing on performance issues pertaining to simulation to better understand the obtrusive program execution events, cache design for servers, and messaging techniques for exploiting the current networking technology were presented. The session was chaired by Mike Nelson of Silicon Graphics.

**SimICS/sun4m : A Virtual Workstation**

Peter S. Magnusson, Fredrik Larsson, Andreas Moestedt, Bengt Werner, Swedish Institute of Computer Science; Fredrik Dalhgren, Magnus Karlsson, Fredrik Lundholm, Jim Nilsson, Per Stentröm, Chalmers University of Technology; Håkan Grahn, University of Karlskrona/ Ronnebäck

Peter Magnusson presented the paper describing the capabilities and the current status of the instruction-set simulator SimICS/sun4m, which has been developed by his research group at the Swedish Institute of Computer Science (SICS) over the past several years.

Simulation is essentially running a program on a simulator on some arbitrary computer that should behave like a program actually running on a specific target computer. Simulation focuses on capturing characteristics like hardware events induced on a target platform during program execution and some details of the software running that are otherwise difficult to gather. Gathering such detailed characteristics using simulators does involve a slowdown of typically two to three orders of magnitude in program execution compared to its execution on native hardware.

System-level simulators facilitate understanding the intricacies of program execution on a target system because of their capability to re-create an accurate and complete replica of the program behavior. Such simulators have thus been an indispensable tool for computer architects and system software engineers for studying architectural design alternatives, debugging, and system performance tuning.

SimICS/sun4m is an instruction-set simulator that supports more than one SPARC V8 processor and is fully compatible with the sun4m architecture from Sun Microsystems. It is capable of booting unmodified operating systems like Linux 2.0.30 and Solaris 2.6 directly from dumps of the partitions that would boot a target machine. Binary compatible simulators for devices like SCSI, console, interrupt, timers, EEPROM, and Ethernet have been implemented by Magnusson's research group. SimICS can extensively profile data and instruction cache misses, translation look-aside buffer (TLB) misses, and instruction counts. It can run realistic workloads like the database benchmark TPC-D or interactive applications such as Mozilla.

A noteworthy application of the SimICS/sun4m platform is its use for evaluating design alternatives for multiprocessors. The evaluation of the memory hierarchy of a shared-memory
multiprocessor running a database application was presented as a case study.

In the presentation the performance of the SimICS/sun4m simulator was demonstrated by comparing the execution time of the SPECint95 programs on the target and host, using the train dataset. The slowdown was in the range of 26-75 over native execution for the test environment chosen.

SimICS/sun4m is available for the research community at <http://www.sics.se/simics>. The presentation slides are available at <http://www.sics.se/simics/usenix98>. Magnusson also welcomed those interested in knowing more about his work to contact him at <psm@sics.se>.

A few interesting questions were asked after the presentation. To demonstrate user and system node debugging, evaluation of Mozilla running on top of Solaris on SimICS had been presented in the talk. In the presentation it was also noted that reloading a page required 214 million SPARC instructions, and about 25% of these were spent in the idle loop. The question was whether it was clear as to why so much time was spent in the idle loop. Magnusson said that the answer wasn’t clear to them, and to get the answers to such questions, they were working on adding high-end scripting tools to their simulator because the current tools are not sufficient for detailed analysis.

In reply to the question of what was the hardest problem to solve in the work, Magnusson said that from an engineering point of view it was the modelling of devices at a level to run real SCSI devices, real ethernet drivers, etc. From the research point of view it was the design of memory fast enough and flexible enough to give one the desired information. In reply to the question on use of interpreters as against real-time code generation, Magnusson said that although the “religious belief” of programmers that real-time code generation was faster held true, he wasn’t aware of any group that had actually implemented it with the desired stability. He added that one of the reasons they are going commercial – Virtutech is the new company their group has founded – was the hope that they will have access to resources required to better address such issues, which are often not feasible in the academic research environment.

High-Performance Caching With The Lava Hit-Server

Jochen Liedtke, Vsevolod Panteleenko, Trent Jaeger, and Nayeem Islam, IBM T.J. Watson Research Center

Jochen Liedtke presented the results of an ongoing experiment at the T.J. Watson Research Center on the architecture design for a high-performance server capable of efficiently serving future local clusters of network computers and other future thin clients (PDAs, laptops, pagers, printers, etc.). The key component in their architecture is a generic cache module designed to fully utilize the available bandwidth.

Liedtke’s group envisions future local networks serving thousands up to hundreds of thousands of resource-poor clients with no or little disk storage. In such scenarios the clients will download a significant amount of data from the server, whose performance can become the bottleneck. They suggest that high-performance customizable servers, capable of handling tens of thousands of transactions per second (Tps) with bandwidths of the order of gigabytes per second will be required.

The basic goals of their research were to find the upper bounds, both theoretical and practical, and to find a highly customizable, scalable architecture for such a scenario.

They based their work on the well-established approach of increasing server performance via efficient cache design. The fundamental idea behind their work is
separating the hit-server from the miss-server. The hit-server is connected to both the pool of clients and the miss-server using different Ethernet connections. There could be several Ethernet cards on the hit-server, each connecting several clients. If the desired object is in the hit-server, it is accessed using standard get and put commands; otherwise the miss-server is signalled.

Because the hit-server is vital for performance, they make it general and policy-free, so that it can adapt to any application. The hit-server allows get/put operation on entire as well as partial objects beside providing support for active objects. The miss handling and replacement policy is handled in the customisable miss-servers. To achieve scalability, it is suggested that multiple customised miss-servers, e.g., fileservers, Web proxy servers, etc., could be implemented. Or more hit-servers can be incorporated in the design to increase the overall cache size. The paper describes the mechanisms that allow the miss-servers to support the desired consistency protocol per object.

Throughputs of up to 624 Mbps are possible using the 1 Gbps PCI bus. But current commercial and research servers still achieve rates up to 2,100 and 700 Tps, respectively, for moderately small 1K size objects. It was demonstrated that the problem was not with the network hardware, but with the memory bus. Thus it is imperative to minimize memory bus access, which slows down the performance. The CPU should limit itself to using the L1 and L2 caches as far as possible. Using lazy evaluation techniques and precompiling packets and preprocessing packet digests can facilitate this. L2 misses can be minimized by proper data structuring. Lava's get performance is 59,000 Tps and 8,000 Tps for 1K and 10K objects, respectively. Liedtke explained that the throughput limit of 624 Mbps suggested in the paper was incorrect because they had based their measurements on the time to transmit a single packet using the PCI bus and had not considered the time interval between the "start transmit" signal to the controller and the start of the transmission, which could be used by some other packet in case of multiple packet transmissions.

A simple application of multiple clients booting by accessing 5M-15M of data over a short interval of five minutes was shown to have an average latency of about 1.5 s for 1,000 clients.

Before concluding, Liedtke put up some open questions. They were whether the hit-server could be used in the WAN environment where different protocols were prevalent, how cache friendly will future applications be and if the system can be customized for them, and whether it will be possible to integrate dynamic applications like databases into the design.

Liedtke concluded by saying that the lessons his group had learned from the implementation were that designing from scratch pays. He also suggested that it is a good strategy to separate the generic-fas-simplify from the customisable-complicated-slower and noted that generality goes with simplicity. He also said that even though an ideal case analysis might be wrong, it is essential, and that designing before implementing should be done whenever possible.

Cheating the I/O Bottleneck: Network Storage with Trapeze/Myrinet
Darrell C. Anderson, Jeffrey S. Chase, Syam Gadde, Andrew J. Gallatin, and Kenneth G. Yocum, Duke University; Michael J. Feeley, University of British Columbia

Darrell Anderson presented a messaging system designed to deliver the high-performance potential of current hardware for network storage systems, including cluster file systems and network memory. They note that the I/O bottleneck arises because disks are inherently slow due to mechanical factors. Very fast networks like Myrinet, on which their work is based, offer point-to-point connections capable of 1 GB/s bandwidths for large file transfers and small latencies of 5-10 microseconds for small messages. The network can instead be viewed as the primary I/O path in a cluster, with the goal of achieving I/O at gigabit network speeds for unmodified applications. By allowing all I/O to/from network memory, the I/O bottleneck can be cheated. Also by pipelining the network with sequential read-ahead, write-behind high bandwidth, file access through the network can be achieved. They rely on the Global Memory Service (GMS) developed at the University of Washington, Seattle, to provide the I/O via the network. Myrinet provides link speeds matching PCI bandwidth, link-level flow control, and a programmable network interface card (NIC), which is vital in their environment. Their firmware runs on the NIC, they modify the kernel RPC, and they treat file and virtual memory systems as an extension of the underlying gigabit network protocol stack. Their firmware and Myrinet messaging system is called Trapeze. They have been able to achieve sequential file access bandwidths of 96 MB/s using GMS over Trapeze.

The GMS system that has been used in Anderson's research lets the system see the I/O through the network. GMS is integrated with the file and VM systems such that whenever a file block or virtual memory page is discarded on a node, it is in fact pushed over the network to some other node, where later cache-misses or page-faults can retrieve it with a network transfer.

In the request-response model on which network storage systems are based, a small request solicits a relatively large page or file block in response. In their work they address the challenges in designing an RPC network storage
and its requirements for low overhead, low latency, and high bandwidth. Support for RPC variations, like multiway RPC for directory lookup and request delegation, is provided. Nonblocking RPC used for implementing read-ahead, write-behind is also supported.

Their Trapeze messaging is reportedly the highest bandwidth Myrinet messaging system. It consists of two parts, the firmware running in the NIC and the messaging layer used for kernel or user communication. It supports IP through sockets as well as kernel-to-kernel messaging and is optimized for both block and page transfers. It provides features for zero-copy communication through unified buffering with the system page frame pool and by using Incoming Payload Table (IPT) to map specific frames to receive into. The key Trapeze data structures reside in the NIC, where they are used by the firmware, but are also accessible to the messaging layer. The Send and Receive Rings in the NIC point to aligned system page frames, which are used to send and receive pages using DMA. These page frames can also be mapped into user space. Particular incoming messages can also be tagged with a token that, when used in conjunction with the Trapeze IPT can deliver the message data into a specific frame. This is used in implementing their zero-copy RPC. Their zero-copy TCP/IP over Trapeze can deliver a highly respectable bandwidth of 86 MB/s for 8 KB data transfers.

They short-circuit the IP layer, which is nevertheless available to user applications over the socket layer, in their integration of RPC with the network interface. This avoids the costly copying at the IP layer in the standard page fetch using RPC over IP.

They report highest bandwidths and lowest overheads using the file mapping mmap system call.

Anderson referred those interested in learning more about their work to their Web site <http://www.cs.duke.edu/ari/trapeze>.

A question was asked as to how IP performance could be improved, which came as a surprise to Anderson, who wasn’t expecting the question and handled it by saying that their MTU size is 8 KB and also page remapping is done to avoid the costly data copying to improve the overall performance. Answering questions on reliability of their RPC and data corruption in the underlying hardware, he said that because they were using Myrinet, which provides a hardware checksum and also link-level flow control, messages are not corrupted or dropped in the network.

Session: Neat Stuff
Summary by Kevin Fu
This session consisted of a collection of interesting utilities. Pei Cao from the University of Wisconsin maintained order as the session chair.

Mhz: Anatomy of a Micro-benchmark
Carl Staelin, Hewlett-Packard Laboratories; Larry McVoy, BitMover, Inc.

Carl Staelin talked about Mhz, a utility to determine processor clock speed in a platform independent way. Mhz takes several measurements of simple C expressions, then finds the greatest common divisor (GCD) to compute the duration of one clock tick.

Measuring a single clock tick is difficult because clock resolution is often too coarse. One could measure the execution time of a simple expression repeated many times, then divide by the number of instructions. However, this too has complications. For instance, a compiler may optimize "a++" run many times in a loop. Moreover, interrupts muddle with the measurements by randomly grabbing CPU time.

Staelin proposed a solution based on ideas learned from high school chemistry and physics to determine atomic weights. Measure the time of simple operations; then use the GCD to determine the duration of one clock tick. Mhz uses nine C expressions for time measurements. The expressions have inter- and intra-expression dependencies to prevent the compiler from overlapping execution of expressions. The operations must also be immune to optimization and be of varying complexity.

Mhz requires the operations to have relatively prime execution times. However, measurements will have variants and fluctuations. Therefore, mhz must minimize noise and detect when a measurement is incorrect. Mhz prunes out incorrect results by measuring many executions of a particular expression. If any particular execution is off by more than a factor of five when compared to other executions, the result is disregarded. Mhz calculates the duration of one clock tick using many subsets of the nine measurements. To produce a final answer, mhz takes the mode of the calculations.

The mhz utility works on x86, Alpha, PowerPC, SPARC, PA-RISC, Linux, HP-UX, SunOS, AIX, and IRIX. Mhz is accurate and OS/CPU independent. Mhz works in Windows NT, but NT does not offer the gettimeofday() call. As a result, Staelin used NT’s native, something-left-to-be-desired timer. Mhz produced correct results, but Staelin did not report this because he does not want to support NT. Porting is painful for a variety of reasons.

Staelin was also asked about loop overhead and interrupts. Mhz was developed with a timing harness that performs a variety of experiments to detect clock granularity. Mhz can remove the overhead caused by the gettimeofday() call. Interrupts are random and hence dealt with by using multiple experiments.

An audience member asked whether mhz could produce more accurate results when given more time to compute. Staelin responded, “Good benchmarking hygiene should be good. We wanted
something that would work in a second or so.”

There may be other areas of computer performance where this method has applicability. This is a trick that can go into your mental toolkit. See <http://www.bitmover.com/> for the source code.

Automatic Program Transformation with JOIE

Geoff A. Cohen and Jeffrey S. Chase, Duke University; David L. Kaminsky, IBM Application Development Technology Institute

Geoff Cohen, an IBM graduate fellow and doctoral student at Duke, talked about load-time transformations in the Java Object Instrumentation Environment (JOIE). Transportable code allows sharing of code from multiple sources. Cohen used JOIE as an environment toolkit to transform compiled Java bytecode.

There already exist binary transformation tools such as OM/ATOM, EEL, and Purify. BIT and BCA allow transformations in Java. However, BCA does not modify bytecodes, and BIT only inserts method calls into bytecodes. Neither is a general transformation tool.

There are a few kinds of transformers. A symbolic transformation could add interfaces or change names, types, or superclasses. A structural transformation could add new fields, methods, or constructors. Bytecode transformation allows for insertion, reordering, or replacement of bytecodes within method implementations. This last transformation significantly distinguishes JOIE from BCA.

Such transformers can extend Java to support generic types and new primitives. For instance, transformers can work with caching, security, and visualization for system integration. Moreover, transformers can add functionality such as versioning or logging.

Load-time transformations with JOIE are incremental, automatic, and universal. JOIE is an enabling technology that gives users more control with programs. Related issues are performance, security, safety, and ease of use.

An audience member asked why not perform transformations in the JIT/JVM. Cohen’s response was this method is not platform independent and is harder to write. In the JIT, symbols may have been lost.

JOIE is written in Java. Performance is on the order of single-digit milliseconds. But once time allows for some tuning, Cohen expects JOIE to run in hundreds of milliseconds.

Responding to a question, Cohen said that it is possible to debug transformed code, but it is very hard. The JVM should prevent anything unsafe created by JOIE at runtime (e.g., read /etc/passwd).

Finally, an audience member asked about reversibility: could a transformation be undone by another transformation. In theory, this is possible, but some functions are one-way.


Deducing Similarities in Java Sources from Bytecodes

Brenda S. Baker, Bell Laboratories, Lucent Technologies; Udi Manber, University of Arizona

Brenda Baker spoke about how to detect similarities in Java bytecode. She is interested in string matching and Web-based proxy services. Java is the juggernaut and is expected to be widespread and ubiquitous. Typically, the bytecode is not distributed with the source code when programmers want to keep the source secret. Baker’s goal is, given a set of bytecode files, to discover similarities among the bytecode files that reflect similarities among their Java source files.

Furthermore, all this should happen without access to Java source files.

Detecting similarities has application to plagiarism detection, program management to find common sources, program reuse and reengineering, uninstallation, and security (similar to known malicious code). For instance, one could detect the incorporation of benchmarks into programs or whether JOIE was applied. There is also a potential battle against code obfuscators.

Baker adapted three tools: siff finds pairs of text files that contain a significant number of common blocks (Manber), dup finds all longest parameterized matches (Baker), and diff is a dynamic programming tool to identify line-by-line changes (GNU).

Siff and diff are not too useful on raw bytecode, even when the byte code is disassembled. When changing a 4 to a 5 in two lines of a 182-line Java file, diff generated 1,100 lines of output on the disassembled bytecode, but siff found less than 1% of similarity.

Baker described three experiments. The first experiment involved random changes to a Java source file (insertion, deletion, substitution within statements). The bytecode was compiled, disassembled, then encoded. The average similarity in the disassembled code using siff never grew larger than 9% off from the same measurement on the Java source. Averages stayed very close, making this a promising approach.

In the second experiment, Baker’s group tried to find similarities in 2,056 files from 38 collections. Thresholds were set as follows: siff reported pairs with at least 50% similarity, dup reported pairs matching at least 200 lines. Nine pairs of programs across different collections were reported as similar by both siff and dup. Eight of these had the same name. One program had the same implementation of the MD5 hash algorithm.

One pair was reported only by dup—probably a false positive. However, siff reported 23 pairs unreported by dup.
Some had similar names while the other pairs consisted of one very small file and one very large file. The small/large file pairs are probably false positives.

Experiment three involved false negatives. Baker's group asked friends to randomly pick 10 programs from set of 765 Java programs. The person would make random changes, then compile the Java code—even with different compiler versions. The bytecode was then returned in random order.

Of the 12 pairs of similar code, siff found nine of 12 pairs with a threshold of 65%; dup found eight pairs with a threshold of 100 lines. Together siff and dup found 10 pairs. There is a trade-off between false positives/negatives and the threshold.

Baker found the offsets to be important for matching. Also, siff can handle large amounts of code, but diff requires the most intensive computation. When analyzing lots of files, first use siff, then dup, then diff. Diff has a quadratic blowup with respect to the number of file inputs.

An audience member asked whether Baker had tried comparing the output of two different compilers. Baker doubts her group will find much similarity. But if you have the code, you could compile in another compiler to test for similarity. As for false positives, if you lower the threshold too far, you could get hundreds of false positives. Moving code around will not affect dup, but will affect siff. This all depends on the threshold. Using siff, dup, and diff in combination makes detection more powerful.

In further research, Baker's group hopes to use additional information in bytecode such as the constant pool.

**Session: Networking**

**Summary by Jon Howell**

The networking session was chaired by Elizabeth Zarchy of Silicon Graphics.

**Transformer Tunnels: A Framework for Providing Route-Specific Adaptation**

Pradeep Sudame and B. R. Badrinath, Rutgers University

Pradeep Sudame presented the concept of transformer tunnels as a way to provide better service to mobile hosts that encounter diverse networks. In a day, a mobile host might access the network at large over a modem, a cellular phone, a wireless LAN in the office, and a high-speed wired LAN. Each network has different properties, and transformer tunnels provide a way to manipulate the traffic going over the mobile host's link to minimize certain undesirable effects.

The mechanics of transformer tunnels are as follows: a routing table entry at the source end of the tunnel indicates that packets bound for a given link should be transformed by a certain function. The source node transforms the packet payload, rewrites the header to point to the tunnel destination, rewrites the protocol number to arrange for the transformation to be inverted at the far end, and attaches the original header information to the end of the packet so it can be reconstructed at the other end.

When the packet arrives at the destination, its protocol number directs it to the appropriate inverse transformation function. The reconstructed packet is delivered to IP, where it is delivered in the usual way to an application on the local host or forwarded on into the network.

Sudame gave interesting examples of how transformer tunnels can provide useful trade-offs for mobile hosts on links with different characteristics. A compression module is useful on slow links, trading off host-processing overhead. A big packet buffer compensates well for links with bursty losses (such as during a cell hand-off), trading off memory requirements. A tunnel that aggregates packets to reduce radio transmitter on-time reduces energy consumption, trading off an increase in the burstiness of the link.

Joe Pruett, in the Q/A period, asked how the transformer deals with a maximally sized packet to which it needs to add overhead. Sudame responded that, for optional optimizations, it would be passed on unchanged; for mandatory transformations such as encryption, it would be transformed and then fragmented.

Ian Vaughn asked if IPsec was used for encryption, to which Sudame replied that they used only a simple exclusive-OR as a proof-of-concept.

Elizabeth Zarchy asked how difficult it was for an unfamiliar programmer to write a transformation function. Sudame replied that it required the programmer to be only somewhat aware of systems programming concepts.

Sudame provided the following URLs for more information and indicated that the group would like many people to try out the code and comment on it.


**The Design and Implementation of an IPv6/IPv4 Network Address and Protocol Translator**

Marc E. Fiu ACS, Vincent K. Lam, and Brian N. Bershad, University of Washington

Marc Fiu ACS discussed an IPv6/IPv4 Network Address and Protocol Translator (NAPT). He identified three possible scenarios in which one might configure a NAPT: use within an intranet, providing your shiny new IPv6 systems with access to the existing IPv4 Internet, and duct-taping your rusty old IPv4 systems to the emerging IPv6 Internet. As he began his
talk, Fiu czynski fumbled with the pointer, but then fell back on his Jedi light saber training, muttering, "Luke, use the laser pointer."

He outlined the project’s goals for a translator: a translator should be transparent so that the end host is oblivious of its presence. It must scale with the size of the network it is serving. It should be failure resilient, in that it can restore service after a reboot. It should, of course, perform suitably. And finally, it should deploy easily.

A translator must attend to several issues. It needs to preserve the meaning of header bits across the header formats. It translates addresses between the IPv4 space and the IPv6 space, which it can do using a stateful or stateless server. And it also needs to translate certain higher-level protocols such as ICMP and DNS that encode IP addresses in the IP payload.

The group built two translators, one stateful and one stateless. The stateful translator has a table of IPv4 to IPv6 address mappings. It attempts to garbage-collect IPv4 addresses to reduce the number needed to serve a site. This garbage collection was challenging because “you might break someone’s ongoing communication... that would be bad... that’s definitely not a goal of the translator.” However, because the translator is stateful, it is not scalable or fault resilient; because it requires rewriting some transport protocol headers, it is not transparent.

The stateless translator uses special IPv6 addresses that map one-to-one with IPv4 addresses. It is scalable, fault resilient, transparent, and has no need to garbage-collect IPv4 addresses. However, using the special compatibility addresses means that routers will still have the “stress” of routing IPv4-like addresses, a problem IPv6 addresses are designed to relieve. Fiu czynski concluded that a stateless translator is best suited to connecting an IPv6 site to the IPv4 Internet or for translating within an intranet.

Joe Pruett asked about DNS translation and whether all internal IPv6 nodes could be reachable from the outside network using IPv4 addresses. Fiu czynski replied that the stateful translator would have to garbage-collect addresses to share them among internal hosts and translate (or directly answer?) DNS queries according to the current mapping.

Greg Minshall asked what the difference was between IPv4 to IPv4 translators and Washington’s IPv6 to IPv4 translators. The reply was that IPv4 NATs are stopgap measures with no clear replacement, but IPv6 translators are a transitional mechanism meant to be eventually removed.

Dave Presotto asked whether the system was rule based, that is, whether he could add new translation functions, other than IPv4 to IPv6 translation, in order to perform other functions using the same system. Fiu czynski expressed confidence that such an extension would be straightforward.

B.R. Badrinath asked if multicast address translation would be a problem, to which Fiu czynski offered a succinct “yes.”

The work is documented at <http://www.cs.washington.edu/research/networking/napt>, and source will be available there soon.

Increasing Effective Link Bandwidth by Suppressing Replicated Data
Jonathan Santos and David Wetherall, Massachusetts Institute of Technology

Jonathan Santos spoke about his group’s work in identifying and suppressing replicated data crossing a given network link. The work applies to any link that is a bottleneck due to cost or congestion reasons. The novel approach of the project was to identify redundancy in packet payloads traversing the link without using protocol-specific knowledge.

Santos defined “replicated data” as a packet payload that is byte-for-byte identical to a previously encountered packet.

The researchers studied a packet trace from an Internet gateway at MIT and discovered that 20% of the outbound volume and 7% of the inbound volume of data met their definition of replicated. HTTP traffic was responsible for 87% of the replication found in the outbound trace, and 97% of the volume of replicated data was delivered in packets larger that 500 bytes, indicating that per-packet compression savings would dwarf any added overhead.

To identify whether the replication could be detected and exploited in an online system, they graphed replicated volume against window size. The graph had a knee at around 100-200MB, signifying that most of the available redundancy could be exploited with a cache of that size.

Their technique for redundancy suppression involved caching payloads at both ends of the link and transmitting a 128-bit MD5 fingerprint to represent replicated payloads. One issue involved retransmitting the payload when the initial packet (containing the real payload) is lost. They also prevent corruption due to fingerprint collisions (the unlikely possibility that two payloads share the same MD5 checksum) in the absence of message loss. (Greg Rose from Qualcomm Australia pointed out that RSA, Inc., issues a cash prize if you discover an MD5 collision. Hopefully, the software reports any collisions to the system administrator.)

Santos concluded that their system was a cost-effective alternative to purchasing link bandwidth and that it complements link-level compression well.

Fred Douglass inquired whether they might be able to identify and compress very similar but not identical packets in an online fashion. Santos suggested using fingerprinting at a finer granularity (over parts of packets).

Nick Christenson pointed out that most of the replication is due to outbound
HTTP traffic and asked whether it might have been nearly as effective to simply use a Web cache on the outbound end of the link. Santos said they assumed client-side and proxy caches were in use when the traces were taken. [This does not account for the redundancy available if all clients passed through the same Web cache at the outbound end of the link, which appeared to be Christenson's point.]

Andy Chu pointed out that, to save bandwidth on a typically congested link to an ISP, one must funnel all data through one link and put the box at your ISP. [Also observe that the cost savings will apply only to the link bandwidth; the ISP will surely still desire compensation for the now-increased use of its upstream link.]

**Session: Security**

Summary by Kevin Fu

The papers in this session dealt with controlled execution of untrusted code. Specifically, the papers discuss how to confine untrusted code to a safe environment. Fred Douglas from AT&T Labs – Research served as the session chair.

**Implementing Multiple Protection Domains in Java**

Chris Hawblitzel, Chi-Chao Chang, Grzegorz Czajkowski, Deyu Hu, and Thorsten von Eicken, Cornell University

Chris Hawblitzel gave a confident, well-paced presentation of the J-Kernel, a portable protection system written completely in Java. The J-Kernel allows programmers to launch multiple protection domains within a single Java Virtual Machine (JVM) while maintaining communication and sharing of objects and classes in a controlled way.

Hawblitzel listed three ways an applet security model can enforce security:

- restrict which classes an applet can access (type hiding)
- restrict which objects an applet can access (capabilities)
- perform additional checks (stack inspection)

However, a problem persists in that applets have no way to communicate in a secure, controlled way. Therefore, the J-Kernel group decided on three requirements for their protection system:

1. Revocation. Java provides no way to revoke references to objects. Therefore, the J-Kernel must provide its own revocation mechanism on top of Java.
2. Termination. If one merely stops a domain's threads, there may still be reachable objects from other domains. Such domains will not be garbage-collected. Therefore, the J-Kernel must free up objects when a domain terminates.
3. Protection of threads. In maintaining control over a thread, ownership must change during a boundary crossing of a cross-domain call. Java lets you stop and change the priority of threads. This could allow for malicious behavior. The J-Kernel should not allow outside changes to a thread when another domain is in control.

The J-Kernel distinguishes between objects and classes that can be shared between domains – and what is private to a single domain. Furthermore, the J-Kernel necessitates a revocation mechanism only for shared objects, simplifies security analysis of communication channels, and allows the runtime system to know which objects are shared.

Hawblitzel noted that it can be hard to maintain a distinction between shared and private information. Private objects must not be passed through method invocations on shared objects to other domains. The J-Kernel solves this by passing shared objects by reference. Private objects passed are by copy.

Using Microsoft's JVM or Sun's JDK with Symantec's JIT compiler on 300MHz Pentium II running Windows NT 4.0, a null J-Kernel local RMI takes about 60x to 180x longer than a regular method invocation. This result is mostly due to thread management and locking when entering a call. Synchronization comprises 60-80% of the overhead. The J-Kernel suffers some performance loss because it is written in Java. See the paper for a table of performance results.

The J-Kernel group created a few sample applications as well. They finished an extensible Web server and are working on telephony server. Private domains interface to the Web, PBX, and phone lines while user domains run servlets to process requests and calls. New services can then be uploaded safely. Related work includes Java sandbox extensions, object references treated as capabilities (e.g., Spin, Odyssey, E), safe language technology (e.g., Java), and capability systems (e.g., Hydra, Amoeba).

One audience member asked how the J-Kernel copies parameters and how it handles data considered to be transient. Hawblitzel explained that the J-Kernel can use serialization to copy objects (the objects are serialized into a big byte array, and then the byte array is deserialized to generate new copies of the objects), or it can generate specialized copy routines that are faster than serialization because they do not go through the intermediate byte array.

Source code and binaries are available for NT and Solaris. For more information, see <http://www.cs.cornell.edu/jkernel/>.
The Safe-Tcl Security Model
Jacob Y. Levy, Laurent Demailly, Sun Microsystems Laboratories; John Ousterhout and Brent B. Welch, Scriptics Inc.

Safe-Tcl allows execution of untrusted Tcl scripts while preventing damage to the environment or leakage of private information. Safe-Tcl uses a padded cell approach (as opposed to pure sandboxing or code signing). Each script (applet) operates in a safe interpreter where it cannot interact directly with the rest of the application. Safe-Tcl's main claim to fame is its flexibility: the capabilities given to the script can be increased or decreased based on the degree to which the script is trusted.

Unfortunately, there was some confusion among the authors about who was supposed to present, with the result that no one showed up at the session. However, the paper is well written and worth the read. You can find related material from Scriptics (http://www.scriptics.com/), the Tcl Resource Center at Scriptics (http://www.scriptics.com/resource/), the Tcl Consortium (http://www.tclconsortium.org/), or the Tcl plugin download page (http://www.scriptics.com/resource/tools/plugin/). The plugin is the best example of an application using Safe-Tcl and is a good starting point for people who want to learn more about Safe-Tcl.

Session: Work-in-Progress Reports

Summaries provided by the authors, and edited and compiled by session chair Terri Watson Rashid

Ray Pereda <rpereda@cs.utsa.edu> talked about a new programming language that he and Clint Jeffery developed at the University of Texas, San Antonio. The language is called Godiva. It is a very high-level dialect of Java.
(http://segfault.cs.utsa.edu/godiva/>

Bradley Kuhn <ckuhn@eb.org> discussed his work at the University of Cincinnati on Java language optimization. The goal of this research is to create a framework for implementation of both static and dynamic optimizations for Java. Such a framework will allow for testing and benchmarking of both new and old dynamic and static optimizations proposed for object-oriented languages in the literature. The framework will build upon existing pieces of the free Java environment, such as Guava and Japhar, to make implementations of new optimizations for Java easy and accessible to all.
(http://www.ebb.org/gnu-spot/)

Jun-ichiro Itoh <itohj@kame.net> talked about his IPv6 and IPsec effort in Japan. The project, called KAME Project, is trying to establish BSD-copylefted, export control-free, reference network code for Internet researchers as well as commercial use. They also intend to incorporate best code for recent topics, such as QoS, IP over satellite, etc.
(http://www.kame.net/project-overview.html)

Oleg Kiselyov <oleg@pobox.com> spoke about an HTTP Virtual File System for Midnight Commander (MC). The VFS lets the MC treat remote directories as if they were on a local filesystem. A user can then view and copy files to and from a remote computer and even between remote boxes of various kinds. A remote system can be an arbitrary UNIX/Win95/WinNT box with an HTTP server capable of running a simple, easily configurable sh or Perl script.
(http://pobox.com/~oleg/USENIX98/)

Ian Brown <iBrown@cs.ucl.ac.uk> described ongoing work on signatures using PGP that can be checked only by people designated by the signer. Typical digital signatures on messages can be checked by anyone. This is useful for contracts, but for confidential messages senders may not want recipients to be able to prove to anyone what they wrote. Comments during the WIP session pointed out that the current approach did not check integrity of encrypted but unsigned data. The authors noted that this is a general PGP problem and have since augmented their design to fix this.

Tom M. Kroeger <tmk@cse.ucsc.edu> from the University of California, Santa Cruz, presented some preliminary work on efficiently modelling I/O reference patterns in order to improve caching decisions. This work is attempting to use models from data compression to learn the relationships that exist between file accesses. They have been addressing the issues of model space and adapting to changing patterns by partitioning the data model and limiting the size of each partition.
They are working to implement these techniques within the Linux operating system.
(http://www.cse.ucsc.edu/~tmk/predictive.html)

Poul-Henning Kamp <phk@freebsd.org> talked about "timecounters," a new concept for tracking realtime in UNIX kernels. With this code NTP can track any current or future possible time signal into the 1E-18 second regime, limited only by hardware issues. A couple of plots showed how NTP tracked a GPS receiver with approximately 10nsec noise.
(http://phk.freebsd.dk/rover.html)

James Armitage <jma@wpi.edu> and Bari Perelli-Minetti <baripm@wpi.edu> spoke about their research with John Rulnick in the Network Operations Research Lab at WPI concerning the causes of soft (transient) errors in workstation and server memory and the effects of these errors on system operation. The techniques being used to explore the effects of soft errors
were also briefly presented. One member of the audience provided information on related experiments on errors occurring in satellite circuits due to cosmic rays in space.

Kostas Magouis (<magouis@eecs.harvard.edu>) briefly talked about his work on eVINO, an extensible embedded kernel for intelligent I/O based on the VINO operating system for task management, extensibility, and networking. He argued that I/O platforms (IOP) in the form of add-on adapters with fast, programmable I/O processors are effective in helping servers face demands of today's gigabit networks and RAID storage systems, offloading interrupt processing and allowing them to better multiplex their resources to application programs. eVINO will focus on I/O and provide extensibility on the IOP with applications such as active networking and Web server caching.

**INVITED TALKS TRACK**

**Repetitive Strain Injury: Causes, Treatment, and Prevention**
Jeff Okamoto, Hewlett-Packard

Summary by Eileen Cohen
Jeff Okamoto, a lively speaker with a sense of humor that helped lighten a grim topic, spoke about Repetitive Strain Injury (RSI) from the depths of personal experience. He worked hard for ten years before his injury — which, he said, may take as long as another ten years to go away — started to occur. At that point he decided to educate himself about RSI, and he used his talk to give the audience the lessons he learned “the hard way — and with ignorance and some amount of fear.”

RSI is a topic of serious import to computing professionals. Not only is more work being done at desktop machines than ever before, but many people are also working longer and longer hours at their computers. It was estimated in 1994 that over 700,000 RSIs occur every year, with a total annual cost of over $20 billion. RSI can be devastating, affecting one’s ability not only to do a job, but also to perform the basic tasks, and enjoy the pleasures, of daily life.

Okamoto began with facts about human anatomy that explain why RSIs occur, then moved on to discuss ergonomics. Companies spend a lot on ergonomics, and even though they’re not all doing it the right way, he urges participating in any ergonomic assessment program your employer offers — “it can be a lifesaver.” He provided detailed tips about hand position and motion, criteria for a good chair, monitor position, and use of pointing devices.

After explaining the range of possible RSI diagnoses and treatments, Okamoto emphasized that if an injury happens on the job, the only way to get proper medical help without paying out of your own pocket is to open a worker’s compensation case. (Many people resist doing this.) Employers are legally bound to provide the necessary paperwork you need to file a claim. Unfortunately, said Okamoto, “having to deal with the worker’s comp system is the worst thing in the world for me.” He gave valuable advice, based on his experience in California, on negotiating the system — in particular how to choose your own physician instead of using one from the short list the state provides, who is “likely to be biased against you.”

“An RSI is something I wouldn’t wish on my worst enemy,” said Okamoto. As a closing note, he raised the specter of what will happen if future computer users, who are getting younger and younger, are not trained as children to type and point properly. “By the time they get out of college, they’ll be 90% on the road to injury.”


**Mixing UNIX and PC Operating Systems via Microkernels: Experiences Using Rhapsody for Apple Environments and INTERIX for NT Systems**


Summary by Kevin Fu

Stephen Walli, vice president of research and development at Softway Systems, started the session by discussing INTERIX, a system to allow UNIX application portability to Windows NT. For the second half of the invited talk, Brett R. Halle, manager of CoreOS at Apple Computer, talked about the Rhapsody architecture. He is the manager of CoreOS.

Walli first noted that INTERIX is the product formerly known as OpenNT. Just this week the product was renamed to avoid confusion with Microsoft products. Walli further noted, “This is not a talk about NT being great.”

Walli explained his “first law” of application portability: every useful application outlives the platform on which it was developed and deployed. There exist migration alternatives to rewriting applications for Win32. For instance, one could use UNIX emulation, a common portability library, the MS POSIX subsystem, or INTERIX. On another side note, Walli exclaimed, “MS POSIX actually got a lot right. Originally, the ttyname function just returned NULL! There were stupid little things, but the signalling and forking were done right.”

However, there is a problem with rewriting an application to Win32. The cost of rewriting increases with the complexity of the application. This led into Walli’s discussion of the design goals of INTERIX:

- Complete porting of runtime environment for UNIX.
- Provide true UNIX semantics for the system services.
- Ensure that changes to application code are made more, not less, portable.
■ Maintain good performance.
■ Do not compromise the security of NT.
■ Integrate INTERIX cleanly into an NT world.

The first big step was implementing Berkeley sockets. This Walli called “a big win.” System V IPC was a big win, too. Other interesting tidbits about INTERIX include:
■ ACLs that map to file permissions
■ no /etc/passwd or /etc/group
■ no superuser

Walli tried not to “mess with the plumbing,” but the INTERIX development team did have to make a /proc system for gdb.

Asked why INTERIX does not implement SUID capabilities, Walli explained that INTERIX did not implement SUID because of implications to the filesystem. If INTERIX provided an interface, it would have to provide complete semantics. As an alternative, INTERIX created a SetUser environment.

Another audience member asked about memory requirements to run INTERIX. Walli noted that NT itself requires more resources when moving from NT 3.51 to 4.0. INTERIX does not need much more space after getting enough memory for NT 32MB is sufficient.

The INTERIX group has ported SSH, but Walli’s CEO got paranoid and said “not in our Web site.” SSH is available in Finland because of export laws.

Walli concluded with his “second law” of application portability: useful applications seldom live in a vacuum. After Berkeley sockets were implemented, the Apache port required just hours. Early porting experiences include 4.4BSD-Lite, GNU source, Perl 5, Apache, and xv. See <http://www.interix.com/>.

For the second half of the session, Halle discussed the Rhapsody architecture. He mainly summarized where Rhapsody came from and what it includes. Rhapsody evolved from NextStep 4.2 (Mach 2.x BSD 4.3, OpenStep) and later became MacOS X (Mach 3.0, Carbon, Blue Box). Portions of the code came from NetBSD, FreeBSD, and OpenBSD. CoreOS provides preemption and protection; supports application environments, processor, and hardware abstractions; and offers flexibility and scalability. Rhapsody runs on Intel boxes as well as the PowerPC. Rhapsody includes several filesystems, such as UFS, ISO 9660, and NFS. However, it does not support hard links in HFS+ or UFS. The networking is based on the BSD 4.4 TCP/IP stack and the ANS 700 Appletalk stack.

The audience then bombarded Halle with questions. Halle said that the yellow box will be available for Win95 and WinNT.

An audience member noted that Halle hinted at the cathedral and bazaar model. Apple could kickstart free software as far as GUIs go. When asked if there are any rumblings about giving back to the community, Halle replied that MKLinux is a prime example.

Another audience participant questioned Rhapsody’s choice to omit some of the most common tools found in UNIX. “If you want to leverage applications, don’t make UNIX shell tools optional,” said the participant. Halle responded with an example where UNIX tools could hurt acceptance by the general population. Your grandma or kid could be using this operating system. Halle reasoned that full UNIX does not make a lot of sense in all environments. You want the right minimal set, but there should be a way to obtain the tools.

Rhapsody does include SSH and TCP wrappers. For more information on Rhapsody, see Programming under Mach by Joseph Boykin et al. or <http://www.mklinux.apple.com/> or <http://www.apple.com/>.

Random, quotable quotation:
Walli: “NT is the world’s fattest microkernel with maybe 36 million lines of code. Now that’s a microkernel.”

Succumbing to the Dark Side of the Force: The Internet as Seen from an Adult Web Site

Daniel Klein, Erotika

Summary by David M. Edsall

Dan Klein

The weather wasn’t the only thing that was hot and steamy during the USENIX conference in New Orleans. In one of the most popular invited talks of the conference, Dan Klein entertained and educated a packed auditorium with his discussion of what is necessary to carry the world’s oldest profession to the realm of the world’s latest technology.

Humorously introduced as a “purveyor of filth” and a “corruptor of our nation’s youth,” Klein went on to show us he was anything but and why everyone around him, including his mother, thinks it is OK. Klein has given talks worldwide and is a freelance software developer. But he is probably best known to the USENIX community as the tutorial coordinator, a skill he used well in teaching all of us everything we always wanted to know about Internet sex but were afraid to ask.

He began by reminding us of the stereotypes of the adult entertainment industry. Images of Boogie Nights, dark alleys, and ladies of the evening all come to mind. What we don’t realize is that there are “ordinary people” working in this industry as well. The owner/administrators of two of the more popular Web sites, Persian Kitty and Danny’s Hard Drive, were each housewives before their online business skyrocketed.
Klein then discussed the two tiers into which the industry can be split. Tier 1 consists mainly of magazine publishers, filmmakers, and prostitutes; while Tier 2 includes resellers such as Klein and phone sex operators. In his explanation of the "product" purveyed by the adult Web business he stated "If there is a phobia, phobia, mania, or pathia for it, it's out there. All the world's queer except thee and me, and I'm not so sure about thee. It's not bad, just different." In his opinion, "You can stick whatever you want wherever you want to stick it so long as what you stick it in wants to get stuck."

How much money can be made from the online sex industry? Examples Klein gave included Persian Kitty, which earned $900,000 in its first year and now is pulling in $1.5 million selling only links. Another company, UltraPics, has 14,000 members at $12 per member. Club Love had 20 times more hits in one day than <www.whitehouse.gov> did in a month. Klein himself is in a partnership of four and his share is up to $3,000 in a good month.

Where does one obtain the product? Some companies simply scan the pictures from magazines in blatant violation of copyright law. Some use Web mirroring to essentially steal their content from similar Web sites. Others, such as Klein's company, download noncopyrighted images from USENET newsgroups and repackage them. Klein's group also provides original content, hiring models and photographers. This comes with its own complications. Klein described the need for qualified photographers, proper lighting, props, costumes, and a director.

Running an adult Web site requires a variety of different technologies. To conserve resources, it helps to use compressed images, and Klein is convinced that the adult industry is one of the major influences driving digital compression. It also helps to split the server load among several machines. He elaborated on a number of ways to accomplish this, including DNS round robin and front-end CGIs. In addition, good programming is useful for automating your site, relieving you of the tedious task of wading through USENET postings, dealing with the administration of members, site updates, logging, reporting, accounting, and checking for people who have shared their passwords, to list a few examples.

Klein discussed, to the dismay of a few members of the audience, ways in which you can boost the visibility of your Web site in top ten lists and search engines. One method uses "click bots" to artificially increase the number of hits your pages receive. Another well-known trick is including a popular word, trademark, or brand in META tags for broader exposure to search engines. Klein also described nastier techniques, such as DNS cache poisoning attacks, misleading ads, and domain name spoofing.

All of this does not come without a price. Klein described the importance of making sure you abide by the law. He described his own methods of USENET scanning as acting as a common carrier, ignorant of whether or not the material is copyright protected, and making sure the images they display carry no copyright notice. When photographing models, his company goes to great lengths to make sure they are of legal age and that they are videotaped to prevent lawsuits. He stressed the importance of reporting all income and paying the IRS their due. Most of all, Klein emphasized, "Don't tempt fate. If they look too young, they probably are."

In the brief question-and-answer session which followed, one attendee asked if the adult Web market has peaked. Klein responded by saying, "I think it can still triple before that happens. It's like the Red Light District in Amsterdam. Eventually, people stopped being really interested, but it still is thriving after years." How will Klein deal with it? "I'm honest and fair and not ashamed of it."

**ADAPT: A Flexible Solution for Managing the DNS**

Jim Reid and Anton Holleman, ORIGIN TIS-INS

Summary by Jim Simpson

As more and more domains and networks come online, the amount of DNS-involved management will only increase. Reid and Holleman implemented a large-scale solution for DNS management for a network in a production environment at Philips. Their presentation was given in two parts: first, Holleman gave a demure explanation about the new DNS architecture, and then Reid gave a sometimes amusing explanation of the tool they developed and the problems they had with deploying it, especially when describing an interaction with a particular NT client.

ORIGIN is a global IT service company created by the merger of two parts of the Philips group. Philips is a large corporation, and with a large, far-reaching, and networked corporation came the need for a large DNS. They use a split DNS policy for security and billing purposes, but the old-style DNS architecture in such an environment grew to the point where zone transfers on the nameservers – making the DNS service erratic – were failing because of resource problems. This created the need and desire to reimplement their DNS architecture; the criteria that had to be met included:

- The systems needed to be adaptable because Philips is diverse in its DNS needs.
- In providing service to Philips, ORIGIN cannot impose a solution – it must fulfill a need.
- Fast lookups are necessary with minimal impact upon WAN.
- The system must be scalable, robust, but at the same time simple, and cannot rely on delegated nameservers 24hours/7days.
They decided to create a centrally managed backbone. They used BSD/OS as their platform because it was already used by large ISPs, the VM subsystem is nameserver-friendly, and it's commercially supported, though some system administrators still pined for Solaris. BIND8 was their choice for software; despite poor documentation they were happy with it during testing. Because BSD/OS runs on i386, there was no real choice in hardware, but this also worked out well due to the low prices associated with Intel-based machines and their ease of replacement. Nameservers with good network connectivity to the global backbone in fixed locations were located and installed in pairs for redundancy.

The actual setup consists of three parts: DNS server architecture, DNS resolver architecture, and DNS content architecture — parts of which were designed to be centrally managed. In order to manage DNS and move it along, they had to come up with a tool which they ended up calling ADAPT. In their scheme, ADAPT eliminates the need for local DNS administration; the local admin controls local data, and the backbone people control the "glue." If local admins update something, they send it to the backbone using dressend. If the data are good, they are put into the DNS database.

There are still some unresolved problems, a few being:
- A lot of sites still run brain-dead nameservers.
- There are strange interactions with WINS.
- There are bizarre nameserver configurations.

However, they met their design goals: costs are low and service is stable.

The session ended with a myriad of questions:
Q: How do you cope with caching nameservers? A: Don't use dynamic templates; use notify instead.
Q: How do you deal with errors in the host file? A: Their makefile creates backups of previous good data; servers are installed in pairs.
Q: How is that done, and what happens when one goes down? A: They're installed manually, and people have to go in and reboot them by hand.

Panel Discussion: Is a Clustered Computer In Your Future?

Moderator: Clem Cole, Digital Equipment Corporation

Summary by Steve Hanson

This panel was interesting in that it presented clustered computing from a number of different viewpoints. It was light on detailed technical information, but made it clear that the panelists, all representing major manufacturers or trends in clustered computing, were looking at the topic from very different viewpoints and were solving different problems. First each panelist presented information on the clustering product produced or used by his company. These presentations were followed by a roundtable discussion of clustering, including a question-and-answer period.

Fred Glover, Digital, introduced the TruCluster system in spring, 1994. TruCluster had limited capabilities at introduction, but now provides a highly available and scalable computing environment. TruCluster supports up to tens of nodes, which may be SMP systems. Normal single-host applications run in this environment. TruCluster provides a single-system view of the cluster. Standard digital systems are used in the cluster and are connected with a high-speed interconnect. A Cluster Tool Kit adds distributed APIs so that applications may be coded to better take advantage of the environment. Digital's emphasis is on running commercial applications in this environment, essentially providing a more scalable computing environment than is available in a single machine as well as providing higher availability through redundant resources.

The primary motivations for HP's cluster environments are to provide higher availability and capacity while lowering the cost of management and providing better disaster recovery. Frank Ho stated that 60% of servers are now in mission-critical environments and that 24x7 operation is increasingly important. Downtime has a significant impact on companies. HP's goal is to guarantee 99.999% uptime per year, which is equivalent to five minutes of downtime per year. Today's high availability clusters guarantee 99.95% uptime and are generally taken down only for major upgrades. HP clearly has emphasized high availability, which is the primary thrust of its marketing to commercial customers.

Bruce Walker's talk was primarily about
the comeback of servers and the evolution of servers from single boxes to clusters. According to Walker, clustering currently falls into categories of clusters providing fail-over (high availability) or NUMA clusters, providing higher cluster performance. Tandem claims that its full SSI (Single System Image) clustering provides both parallel performance and availability. Tandem currently ships 2-6 node clusters, having up to 24 CPUs. Tandem is working with SCO on its operating system, which is based on UNIXware.

The Microsoft agenda for clustered computing is to introduce clustered computing technology in stages. Clustered computing currently is a very small portion of the marketplace. According to Rick Rashid, the current thrust of Microsoft’s strategy is to work on high availability solutions (which are available currently in NT) and to introduce scalability of clustering in 1999.

Ron Minnich spoke about the implementation of high-powered computing clusters on commodity PC systems. There is a history of implementing high-powered computer clusters on commodity systems. FERMILAB and other high-energy physics sites have for years done the bulk of their computing on clusters of small UNIX workstations. The availability of free, stable operating systems on very inexpensive hardware has allowed the design of very high powered computing clusters at comparatively low prices. Minnich made the disputable statement that PC/Open UNIX computing is about 10 times as reliable as proprietary UNIX systems and 100 times as reliable as NT. Having formerly been an administrator for large computer clusters on proprietary UNIX systems at FERMILAB, I somehow doubt that, because the UNIX clusters we were using almost always failed due to hardware failure, not OS failure. I find it difficult to believe that $1,000 PC systems are more than 10 times more reliable than entry-level UNIX desktops. However, I think the point is well taken that this is a means of building reliable computer engines of very high power at a very low price. Redhat currently offers a $29.95 Beowulf cluster CD that includes clustering software for Linux.

A question-and-answer period followed the presentations. The questions indicated that may organizations in the real world are asking for more from the clustering vendors than they are currently providing. Questions were asked about a means of developing and debugging software for a high-availability environment and about how to establish a high-availability network across a large geographic area. The response of the panel members seemed to indicate that they hadn’t gotten that far yet.

Other discussions involved recommendations of other approaches to clustering, including use of Platform Computing’s LSF software product <http://www.platform.com> as well as the University of Wisconsin’s Condor software, which excels for use in environments where unused hours of CPU on desktop systems can be harvested for serious compute cycles <http://www.cs.wisc.edu/condor>.

Berry Kercheval asked whether it is important for a cluster to provide a single system image. As an example he mentioned Condor computing, which is not a single system image design, but provides an environment that looks like a single system to the application software. He also made the point that SSI clusters are likely to be more viable because they are a simpler mechanism for replacing mainframes in a commercial environment.

The Future of the Internet
John S. Quarterman, Matrix Information and Directory Services (MIDS)

Summary by David M. Edsall
Death of the Internet! Details at eleven!
That may be the conclusion you would have drawn had you read Bob Metcalfe’s op-ed piece in the December 4, 1995, issue of InfoWorld <http://www.infoworld.com> where he predicted the collapse of the Internet in 1996. Fortunately, his dire predictions did not come true. In his talk in New Orleans, Internet statistician John Quarterman showed us why.

Quarterman is president of Matrix Information and Directory Services (MIDS) in Austin, Texas, a company that studies the demographics and performance of the Internet and other networks worldwide. Drawing upon the large collection of resources and products available from MIDS, Quarterman educated an attentive audience on the past and current growth of the Internet and other networks before taking the risk of making predictions of his own. (The slides presented by Quarterman, including the graphs, are available on both the USENIX ’98 conference CD and at <http://www.mids.org/conferences/usenix>.)

Quarterman began his talk by discussing the current number of Internet hosts worldwide. Not surprisingly, most of the hosts are located in the industrialized countries, with a dense concentration in large urban centers. What is exciting is the number of hosts popping up in some of the more remote areas of the world. As Quarterman said, “Geographically, the Internet is not a small thing anymore.”

Next, he discussed the history of the growth of computer networking from the humble beginnings of the ARPANET with two nodes at UCLA and SRI, through the split of the ARPANET in 1983 and the subsequent creation of the NFSNET, to the eventual domination of
the Internet over all of the other networks. Quarterman showed how many of the other networks (UUCP, BITNET, and FIDONET) have reached a plateau and are now declining in use while the Internet continues to increase in number of hosts at an exponential rate. He similarly showed the parallel growth in the number of Matrix users (users who use email regardless of the underlying network protocol used for its transmission), with the Matrix users increasing more rapidly due to the multiplicity of networks available in the 1980s.

Quarterman next showed the growth of the Internet in countries worldwide. As expected, the United States currently leads the rest of the world in total number of hosts and has a growth rate similar to that of the Internet as a whole. He attributed the slow growth of the number of Japanese hosts to the difficulty that Japanese ISPs had in obtaining licenses, a restriction that was eased in 1994, leading to a large spurt in the growth rate in Japan now.

Moving on to the present day, Quarterman displayed an interesting plot that reflects the latency vs. time of various hosts on the Internet. It was this graphic that persuaded Bob Metcalfe that large latencies do not remain constant, and hence there will be no global breakdown. (But Metcalfe still maintains that there could be a local crash; this is a much less controversial position, because few people would disagree that there are often localized problems in the Internet.) Even more interesting was an animated image of a rotating globe with growing and shrinking circles representing latencies in various parts of the world during the course of a day. This image showed that the latencies undulate on a daily basis much like the circadian rhythm obeyed by the human body. The image also shows different patterns, depending on which country you study.

Latencies appear to increase at noon in Japan and decrease in the afternoon in Spain.

With the past and the present behind, what lies in the future for the Internet? Using a deliberately naive extrapolation of the data presented, Quarterman predicted that, by the year 2005, the number of hosts in the world will nearly equal the world’s population. He also predicted that the US will continue to have the dominant share of the world’s Internet hosts, but will eventually reach a saturation point. But it is difficult to say where any bends in the curve will come.

His confidence in his projected growth of the Internet is based partly on comparisons he has made between the Internet and other technologies of the past. He presented a plot of the number of telephones, televisions, and radios in the United States vs. time alongside the growth of the world’s population and the growth of the Internet. The growth of the Internet has been much faster than the growth of any of these industries. All three of the older technologies eventually leveled off and paralleled the world’s population growth, whereas the Internet shows no signs of doing so soon. He has yet to find any technology whose growth compares with that of the Internet. At this point, Quarterman asked the audience to suggest other technologies whose growth could be compared to the Internet. Ideas, both serious and humorous, included the sales of Viagra, production of CPU chips, automobile purchases, and size of the UNIX kernel. A grateful Quarterman stated that no other audience had ever given him so many suggestions.

Quarterman finished by discussing possible future problems that may hinder the growth and use of the Internet. In a scatter plot of the number of hosts per country per capita and per gross national product, he showed the audience that the growth of the Internet is also dependent on economic and political conditions around the world. Countries with large numbers of hosts tend to have higher standards of living and less internal strife. He also stressed the importance of the social behavior of those using the Internet. In a long discussion of spam, Quarterman revealed that he prefers that the governments of the world adopt a hands-off approach, leaving the policing of the net to those who have the ability to control mail relays. You can find more information at [http://www.mids.org/span/].

Will the Internet eventually collapse and fold? Stay tuned.

**FREENIX TRACK**

**Machine-Independent DMA Framework for NetBSD**

Jason R. Thorpe, NASA Ames Research Center

Summary by Keith Parkins

Jason Thorpe spoke about the virtues of and reasons for developing a machine-independent DMA framework in the NetBSD kernel. The inspiration seems fairly obvious: if you were involved with porting a kernel to different architectures, wouldn’t you want to keep one DMA mapping abstraction rather than one per architecture? Given the fact that many modern machines and machine families share common architectural features, the implementation of a single abstraction seemed the way to go.

Thorpe walked through the different DMA mapping scenarios, bus details, and design considerations before unveiling the bus access interface in NetBSD. A couple of questions were asked during the session, but most of the answers can be found in the paper.

Thorpe seems to have followed the philosophy of spending his time on sharpening his axe before chopping at the tree. He noted that while the implementation of the interface took a long time, it
worked on architectures with varying cache implementations such as mips and alpha without hacking the code.

Examples of the front-end of the interface can be found at: http://ftp.netbsd.org/pub/NetBSD/NetBSD-current/src/sys/dev/ in the pci, isa, isa, and tc folders. For examples of the backend, look in the ic folder.

**Linux Operating System**

Theodore Ts'o, MIT; Linus Torvalds, Transmeta

Summary by Keith Parkins

They had to take down the walls that separated the Mardi Gras room into three smaller cells for the Linux state of the union talk. Instead of being flanked off into seats that would put them out of visual range, audience members chose to position themselves against the back wall for a closer view. This was not what the speakers had expected because they had initially envisioned the talk being a BOF.

Theodore Ts'o began the aforementioned “state of the union.” He started out by citing a figure gathered by Bob Young, CEO of Red Hat Linux, 18 months earlier. The figure in question was the number of Linux users at that time, which is not an easy figure to derive because one purchased copy of Linux can sit on any number of machines and people are also free to download it. The best results derived from various metrics showed users at that time to number between three and five million. Today, similar metrics show this number at six to seven million, although Corel, in its announcement of a Linux port of Office Suite, claimed the number to be eight million.

Because of these rising figures and subsequent rising interest by commercial developers, Ts'o noted that the most exciting work in the Linux universe was taking place in userland, not in the kernel, as had historically been the case. Ts'o noted the development of the rival desktop/office environments, KDE and GNOME, and new administration tools that make it easy for “mere mortals” to maintain their systems. Ts'o also talked briefly about the Linux Software Base, an attempt to keep a standard set of shared libraries and filesystem layouts in every Linux distribution so that developers don’t lose interest due to porting their software to each distribution of Linux.

Ts'o then spoke about the ext2 filesystem. The first thing he emphasized was that although most distributions use ext2fs, it is not the only filesystem used with Linux. He touched upon ideas such as using b-trees everywhere to show the other work out there. While work continues on ext2fs, Ts'o stated that their number one priority is to ensure that any new versions do not endanger stored data. This means extensive testing before placing the filesystem in a stable release, probably not until 2.3 or 2.4. Features to come include metadata logging to increase recovery time, storing capabilities in the filesystem (a POSIX security feature to divide the setuid bit into discrete capabilities), and a logical volume manager.

When Linus Torvalds took the floor, he too expressed his hope that the presentation would be a BOF. In keeping with this hope, he kept his portion of the state of the Linux union brief before opening the floor to questions. Instead of focusing on what people can expect in future releases, he concentrated on two differences between 2.2 and earlier releases. He quickly noted that the drop in memory prices had encouraged the Linux kernel developers to exploit the fact that many machines have a lot more memory than they used to. He elaborated on the kernel will still be frugal with memory resources, but that it seemed poor form to not exploit this trend. He then noted that although earlier releases had been developed for Intel and Alpha machines, 2.2 will add Sparc, Power PC, ARM, and others to the list. As Torvalds puts it, “anything that is remotely interesting, and some [things] that aren’t” will be supported.

There were many good questions and answers when the floor was opened. When asked if Torvalds and company would make it easier for other flavors of UNIX to emulate Linux, Torvalds replied that although he was not trying to make matters difficult for others, he was not going to detract time from making clean and stable kernels to make Linux easier to emulate. He also noted that the biggest stumbling block for others in this task would be Linux’s implementation of threads.

On a question concerning his stance on licensing issues, Torvalds stated simply that he is developing kernels because it was what he enjoys doing. He went on to state that he personally does not care what people do with his end product or what they call it.

When asked about the Linux kernel running on top of Mach, Torvalds stated that he feels the Mach kernel is a poor product and that he has not heard a good argument for placing Linux on top of it. At one point, Torvalds had thought about considering the Mach kernel as just another hardware port. He said that he later changed his mind, when he saw that the kernel running natively on an architecture runs much faster. This initial question led to a question as to whether Linux will become a true distributed operating system. Torvalds stated that he feels it does not make sense to do all the distribution at such a low level and that it makes more sense to make hard decisions at a higher level with some kernel support.

The closing comment from the floor was a thank-you to Torvalds for bringing the world Linux. Torvalds graciously responded by saying that while he is happy to accept the thanks, he was not as involved in the coding process, and the thanks should go out to all the people involved with writing the kernel and applications as well as himself.
Panel Discussion: Whither IPSec?

Moderator: Angelos D. Keromytis, University of Pennsylvania
Panelists: Hugh Daniel, Linux FreeS/WAN Project; John Ioannidis, AT&T Labs – Research; Theodore Ts'o, MIT; Craig Metz, Naval Research Laboratory.

Summary by Kevin Fu

Angelos Keromytis moderated a lively discussion on IPSec’s past, present, and future. In particular, the panelists addressed problems of IPSec deployment. The panel included four individuals intimately involved with IPSec. IPSec is mandatory in all IPv6 implementations.

A jolly John Ioannidis claims to have been involved with IPSec “since the fourteenth century.” He became involved with IPSec before the name IPSec existed. As a graduate student, Ioannidis spoke with other security folks at an IETF BOF in 1992. Later, Matt Blaze, Phil Karn, and Ioannidis talked about an encapsulating protocol. Finally, to win a bet, Ioannidis distributed a floppy disk with a better implementation of swIPe, a network-layer security protocol for the IP protocol suite.

Theodore Ts'o pioneered the ext2 filesystem for Linux, works on Kerberos applications, and currently is the IPSec working group chair at the IETF. He gave a short “this is your life” history of IPSec. After the working group formed in late 1993, arguments broke out over packet formats. However, the hard part became the management of all the keys. Two rival solutions appeared: Photurus by Phil Karn and SKIP by Sun Microsystems. SKIP had a zero round-trip setup time, but makes some assumptions that were probably not applicable for wide-scale deployment on the Internet. Then ISAKMP Oakley developed as a third camp and was adopted by the NSA. Ironically, the ISAKMP protocol was designed to be modular, but the implementations are not so modular. Microsoft did not implement modularity in order to make the software more easily exportable. Ts'o describes the current status of IPSec as the “labor” phase for key management and procedural administration. Looking to the future, Ts'o notes there is some interest in multicast. But he worries about the trust model of multicast—if 1,000 friends share a secret, it can’t be all that secret. Ts'o also stresses the difference between host- and user-level security. Are we authenticating the host or user? Will IPSec be used to secure VPNs and routing or the user? Right now the answer is VPNs.

Hugh Daniel is the “mis-manager” for a free Linux version of the Secure Wide Area Network (FreeS/WAN). Because of the lords in DC, foreigners coded all of FreeS/WAN outside the US. There is a key management daemon called Pluto for ISAKMP Oakley and a Linux kernel module for an IPSec framework.

Craig Metz then gave a short slide presentation on NRL’s IPSec implementation. Conference attendees should note a change in slide 7 on page 120 of the FREENIX proceedings: it now supports FreeBSD and OpenBSD.

Keromytis opened with a question about deployment. People went through lots of trouble to get IPSec standardized. What are the likely main problems in deployment and use of IPSec? Metz answered that getting the code in the hands of potential users is the hardest part. IPSec does not have to be the greatest, but it has to be in the hands of the users. IPSec does not equal VPN. IPSec can do more and solves real-world problems. Ioannidis commented that the problem with IPSec is that some people want perfection before releasing code. If only three people have IPSec, it is not too useful. This is just like the fax—you need a pool of users before IPSec becomes useful. Key management is also a hard problem.

The next question involved patches. Are patches accepted from people in the US? Daniel replied that you can whine on the bug mailing list, but you cannot say what line the bug is on or what the bug is. Linux FreeS/WAN will not take patches from US citizens. Ts'o explained that MIT develops code in the US, but does not give permission to export. When Ts'o receives a patch from Italy, he is not able to tell if it came from a legal export license. Besides, no one would commit such a “violent, despicable act.”

Metz was asked why the government is interested in IPSec. Metz answered that many people in the government want the ability to buy IPSec off the shelf. A lot of the custom-built stuff for the government leaves something to be desired. Metz further explained, “If we can help get IPSec on the street, the government can get higher quality network security software for a lower cost.”

Ts'o said that Microsoft NT 5.0 is shipping with IPSec. Microsoft wants IPSec more than VPNs. Interoperability with the rest of the world will be interesting. Microsoft has a lot of good people; UNIX people should hurry up.

An audience member asked about the following situation: let a packet require encryption to be sent over a link. Is there a defined ICMP packet that says “oops, can't get encryption on this link”? What is the kernel supposed to return to when this happens? Daniel reported that this is not properly defined yet. Metz expanded that there is a slight flame war now. Some believe this would allow an active attacker to discover encryption policies. Should such a mechanism exist? The answer is likely to be “maybe.”

Ts'o responded, “Think SYN flood. Renegotiating allows for denial of service. This is not as simple as you might think.” Daniel substantiated this with figures for bulk encryption. When you deal with PKCS and elliptic curves, encryption can take a 500MHz alpha to its knees. It could take five minutes! Metz mentioned a hardware solution for things such as modular exponentiation.
Dummynet and Forward Error Correction
Luigi Rizzo, Dip. di Ingegneria dell’Informazione – Università di Pisa

Summary by Jason Peel

Luigi Rizzo took the floor and immediately asked the audience: “how [may we] study the behavior of a protocol or an application in a real network?” His answer? A flexible and cost-effective evaluation tool dubbed “dummynet,” designed to help developers study the behavior of software on networks.

In the scrutiny of a particular protocol or application, simulation is often not plausible — it would require building a simulated model of the system whose features may not even be known. Alternatively, research on an actual network might be plagued as well, perhaps due to the proper equipment not being available, or difficulties in configuration. The solution presented in dummynet combines the advantages of both model simulation and actual network test beds.

With a real, physical network as a factor in the communication of multiple processes, traffic can be affected through propagation delays, queuing delays (due to limited-capacity network channels), packet losses (caused by bounded-size queues, and possibly noise), and reordering (because of multiple paths between hosts). These phenomena are replicated in dummynet by passing packets coming in to or going out of the protocol stack through size-, rate-, and delay-configurable queues that simulate network latency, dropped packets, and packet reordering. Dummynet has been implemented as an extension of the ipfw firewall code so as to take advantage of its packet-filtering capabilities and, as such, allows configuration that the developer may already be acquainted with.

The other tool Rizzo presented is an implementation of a particular class of algorithm known as an erasure code. Erasure codes such as his are used in a technique called Forward Error Correction (FEC) as an attempt to eliminate the need for rebroadcasts caused by errors in transmission. In certain situations, particularly asymmetric communication channels or multicast applications with a large number of receivers, FEC can be used to encode data redundantly, such that the source data can be successfully reconstructed even if packets are lost. As useful as this may seem, however, FEC has only rarely been implemented in network protocols due to the perceived high computational cost of encoding and decoding. With his implementation, Rizzo demonstrated how FEC can be taken advantage of on commonly available hardware without a significant performance hit.

To develop this linear algebra erasure code, known technically as a “Vandermonde” code, Rizzo faced several obstacles. First, the implementation of such a code requires highly precise arithmetic; this was solved by splitting packets into smaller units. Second, operand expansion results in a large number of bits; by performing computations in a “Finite” or “Galois” field, this too was overcome. Lastly, a systematic code – one in which the encoded packets include a verbatim copy of the source packets — may at times be desired so that no decoding effort is necessary in the absence of errors. By using various algebraic manipulations, Rizzo was able to obtain a systematic code.

Nearing the close of his session, Rizzo utilized a FreeBSD-powered palmtop to demonstrate the ease of use with which dummynet can simulate various network scenarios. Then he used this virtual test bed network as he showed RMDP (a multicast file-transfer application making use of FEC) in action. The crowd was enthused, and Rizzo was let off the hook with only one question to answer, “Where can we get this?”

Arla – A Really Likable AFS-Client
Johan Danielsson,
Parallelldatorcentrum KTH; Assar Westerlund, Swedish Institute of Computer Science

Summary by Gus Shaffer
Johan and Assar gave a very exciting presentation about their new, free, and portable AFS client, Arla, which is based on the Andrew File System version 3.

A major part of their presentation explained the difference in structure between Transarc’s AFS and Arla. Arla exports most of its internals to a highly portable user-land daemon, as opposed to Transarc’s massive kernel module. The presenters admitted that their change did bring up some performance issues, but the speed of porting was largely increased: they already support six platforms, with four more on the way.

Johan and Assar also mentioned that students at the University of Michigan’s Center for Information Technology Incorporation have incorporated disconnected client modifications originally written for AFS, and they hope to eventually incorporate this code into the main Arla source tree.

The most exciting announcement of the presentation concerned the other half of client-server architecture — the developers have an almost-working fileserver! They presently need to add database servers (volume server and protection server) to have a free, fully functional AFS server.

The presentation drew questions from such noted people as CITI’s Jim Rees and produced tongue-in-cheek comments along the lines of, “Production use means someone bitches wildly if it doesn’t work.”

October 1998 logrin
ISC DHCP EDistribution
Ted Lemon, Internet Software
Consortium

Summary by Branson Matheson
Ted Lemon gave a good talk regarding his
Dynamic Host Configuration Protocol
implementation. About 50 people attend-
ed the discussion. He described the ben-
efits of a DHCP server, which include
allowing users plug-and-play ability,
making things easier for network/sysad-
mans regarding address assignment, and
how conflicts are prevented. He also dis-
cussed potential improvements for ver-
sion 3, including authentication,
Dynamic DNS, and fail over protocol.
Lemon also mentioned that ISC is part of
his implementation and that it is assisting
with standards and some financing. The
question-and-answer session was full of
good comments and discussion. There
was quite a bit of talk about the different
ways people have implemented a dynam-
ic DNS setup, how to id the client
requesting an ip.

Heimdal – An independent implementa-
tion of Kerberos 5
Johan Danielsson,
Paralleladatorcentrum KTH; Assar
Westerlund, Swedish Institute of
Computer Science

Summary by Branson Matheson
Johan Danielsson and Assar Westerlund
travelled from Sweden to discuss their
implementation of a free Kerberos soft-
ware. Heimdal, which is named after the
watchman on the bridge to Asgard, was
developed independently and is interna-
tionally available. They described
Kerberos/Heimdal in general and then
also discussed some of the additions and
improvements that they had made
including 3DES, secure X11, IPv6, and
firewall support. They also discussed
some of the problems they had with the
implementation and how they solved
them, including how to get secure packets
across a firewall. During the question-
and-answer session, there was lots of dis-
cussion of the S/Key and OPIE, encryp-
tion, and proxy authentication. Although
the language barrier sometimes seemed to
be a factor, this discussion went well
and was well received.

Samba as WNT Domain Controller
John Blair, University of Alabama

Summary by Steve Hanson
John Blair is the primary author of the
elegant documentation for the freeware
Samba package. I find this interesting
because the Samba documentation is not
only a fine example of how good the doc-
umentation for a freeware package can
be, but also one of the best sources for
information on how Microsoft's
Win95/NT file sharing works.
Unfortunately, Blair was so busy working
on documentation for the Samba project
and his new book on Samba (Samba: Inte-
rating UNIX and Windows) that he
did not complete a paper for the confer-
ence. Therefore the talk didn't relate to
a paper, but was a more general discussion
of Samba's progress toward having
domain controller capabilities, as well as
Samba development in general. This was
a very interesting talk, more as a general
viewpoint on the motivations of the Samba
project than anything else.

Due to the extra time created by the
missing first presentation, Blair chose to
have a question-and-answer period
before beginning his talk. Many questions
were asked, primarily about trust rela-
tionships with NT servers and using
Samba as a domain controller. Blair
responded that although the code for
domain control exists in the current
Samba release, it isn't considered produc-
tion quality. Many sites are successfully
using the current code, however. In addi-
tion to some details on how to enable the
domain controller code in Samba, Blair
presented some information on the diffi-
culty of creating interoperability between
the NT world and UNIX. This includes
mapping 32-bit ID to UNIX IDs, having
to develop by reverse engineering, bugs in
NT that cause unpredictable behavior,
etc. The inevitable discussion of NT secu-
ritv was interwoven into the talk, particu-
larly in regard to new potential security
issues and possible exploits that were dis-
covered while reverse engineering the
domain controller protocols.

The balance of Blair's talk was devoted to
the Samba project in general. Several
issues about the need for Samba were
raised. Although there are a number of
commercial software packages allowing
SMB file sharing from UNIX, Samba
holds a unique place in the world because
the code is freely available and well docu-
mented. Samba code and documentation
are the best window into determining
how Microsoft networking actually
works. In some cases bugs and potential
exploits have been discovered, some of
which Microsoft has fixed. Public scruti-
ny of the NT world is possible only
through projects such as this. It seems
unlikely that corporate America will learn
to say no to the Windows juggernaut, but
at least this sort of review stands some
chance of opening the Windows world of
networking to review.

Samba is also important because the
UNIX platforms on which it runs are
more scalable than the current NT plat-
forms. The release dates for NT 5.0 and
Merced processors seem to be continually
moving over the horizon, so interoper-
able UNIX platforms at least offer a scal-
able stable place to host services in the
meantime. The work being done here
also raises the interesting prospect of
being able to administer the NT world
from the UNIX platform.

Further information on Samba is avail-
Using FreeBSD for a Console Server
Branson Matheson, Ferguson Enterprises, Inc.

Summary by Branson Matheson
Branson Matheson gave a discussion of his implementation of a console server using FreeBSD. He described the hardware and software requirements and the problems associated with the installation of a console server. The problems included security concerns, layout, and planning. He went into specifics over the implementation of the software and hardware. There was some good discussion about other implementations during the question-and-answer session. Security seemed to be the central theme of the questions: maintaining the security of the consoles while giving the system administrator the necessary privileges and functionality.

CLOSING KEYNOTE SPEECH
Reconfigured Self as Basis for Humanistic Intelligence
Steve Mann, University of Toronto
Summary by Jim Simpson

Steve Mann

As we spin and hurtle ourselves faster toward the future, we find the tools helping us can now be used against us in a myriad of ways. Steve Mann offered a very sharp, pointed, and humorous presentation about taking technology back, through Humanistic Intelligence.

Humanistic Intelligence is the interaction between a human and a computer, and encapsulates the two into a single functioning unit. The ideal is for the computer to augment the reality of the human working with it. It sounds like that goal is well on the way; Mann typed most of his thesis while standing in line, noting that his primary task was to stand and wait in a line, but that WearComp, his implementation of Humanistic Intelligence, allowed for a secondary task where he could be creative. WearComp consists of a host computer on the person's body, a pair of customized glasses and connectivity; specifics about WearComp are at <http://www.wearcomp.org/wearhow>. Note that WearComp runs on an OS, and not a virus. Despite large evolutionary strides, Mann commented about the setup, "The problem with wires is you get tangled up in them."

A few of the more interesting scenarios and uses of WearComp include visual mediation. Say you don't wish to see a particular ad. You have your image processor map a texture over it. Imagine if you were about to be mugged on the street. You could simply beam back an image of the perpetrator. Finally, and perhaps one of the most important uses is that people could better understand each other. Mann illustrated this with a story about being late. Whoever is waiting could simply see the world through your eyes and, instead of being suspicious or upset, know the person is being genuine with the explanation.

We then were treated to an excerpt from a video documentary Mann did, called Shooting Back. It demonstrated the modern double standard we're held to; as Mann asked about surveillance cameras in everyday stores, he was bounced from person to person. Mann turned the tables, and when those persons were asked how they felt being videotaped, they had the same reaction that prompted Mann's deployment of a video camera. What's more interesting is that while pretending to initiate recording the other party, he'd been surreptitiously recording everything with WearComp.

Toward the end of the session, the chair began to check his watch nervously; it seemed almost awkward when the chair had to tell Mann the time because Mann was well aware of the time — it was happily ticking away in the form of an xclock on the other side of his glasses.

Because this is a working product, Mann answered a few questions about WearComp and how it has fared: What operating system do you use? Linux 2.0, RedHat, but Mann has written custom stuff like DSP. Has the system ever cut out? Yes, there are dark crashes. The most common thing is for the battery to die, but there is a 30-minute warning system. You don't want to be in mediated reality, walk across the street, and then have a system cut out the moment a truck is barreling toward you. Can you show us what you're seeing? No, the video output is in a special format that won't hook up to a standard VGA projector.

Free Stuff
Opinion by Peter H. Salus

[Editor's Note: While Peter is director pro tem of the Tcl/Tk Consortium, he is not an employee of any of the companies mentioned in this report.]

The Association held its June meeting in hot, steamy New Orleans. I emerged from the hotel into the humid heat only twice in four days. Inside the hotel it was cooler and there were lots of folks to talk to.

However, for the first time in a dozen years, I hardly attended any mainline technical papers; I went to the parallel FREENIX track. I learned about NetBSD, FreeBSD, Samba, and OpenBSD. I went to the "Historical UNIX" session (it's 20 years since Dennis Ritchie and Steve
Johnson ported V7 to the Interdata 8 and Richard Miller and Juris Reinfieldds ported it to the Interdata 7) and to the 90-minute history BOF that extended to nearly three hours. And I was present at the awards, the Tcl BOF, Linus Torvalds’s talk, and James Randi’s entertaining, but largely irrelevant, keynote.

There was also a session on Eric Raymond’s “The Cathedral and the Bazaar,” which was largely a love-in conducted by Eric and Kirk McKusick until the very last minutes, which were occupied by a lengthy flame from Rob Kolstad. More heat was radiated than light was shed.

If you know me, you will see the connecting motif: ever since I saw UNIX in the late 1970s, I have been interested in the way systems develop: in Raymond’s terms, I’m much more a bazaar person than a cathedral architect. (And remember that although treasures can be found in a bazaar, Microsoft products are misshapen in a cathedral in Washington.)

UNIX was the first operating system to be successfully ported. And it was ported to two different machines (an Interdata 7 and an Interdata 8; later Perkin-Elmer machines) virtually simultaneously independently by teams half the planet apart. Not only that, but V7 contained awk, lint, make, uucp (from Bell Labs); Bill Joy’s changes in data block size; Ed Gould’s movement of the buffers out of kernel space; John Lion’s (New South Wales!) new procedure for directory pathnames; Bruce Borden’s symorder program; etc., etc. A bazaar stretching from Vienna through the US to Australia. I have outlined the contributions to the development of UNIX in several places, but the important thing is to recognize the bazaarlike activity in the 1970s and 1980s. With Linux, we progress into the 1990s.

NetBSD, OpenBSD, FreeBSD, BSDI, SVR5, and the various Linuces are the result of this bazaar, with AT&T, Bell Labs, Western Electric, UNIX International, X/Open, OSF, and (now) the Open Group flailing about to get the daemons back in the licensing box. No hope.

John Ousterhout (who received the annual Software Tools User Group Award) nodded at both open development and Eric Raymond at the Tcl BOF, saying that he was slightly toward the cathedral side of the middle. By this he meant that he welcomed extensions and applications to Tcl/Tk, but that he reserved the right to decide what was included in any “official” distribution. Because Ousterhout is an intelligence whom I would entrust with such a role, I foresee no problem. But what if Tcl were usurped by an evil empire?

Cygnus, RedHat, Walnut Creek, Scriptics, etc., are examples that money can be made from “free” source. (This is blasphemy to “pure” FSFers, who think that the taint of, say, MetroX in RedHat’s Linux distribution poisons all of RedHat. They’re extremists.) Integrating free software with solid, useful proprietary software is a good thing: it tests the proprietary software among the wider user community, and it spreads the free software to the users of the proprietary stuff.

This aside, I thought the two papers on IPSec (by Angelo Keromytis and Craig Metz) were quite good. Thorpe on NetBSD and de Raadt on OpenBSD were quite lucid, as was Matheson on FreeBSD. Blair on Samba was as good as I had hoped. Because the other author in the session was a no-show, we had an open Q&A and discussion for nearly 90 minutes.

Microsoft may control 90% of the world’s desktops, but all the important developments in OSes are clearly taking place in the bazaar.
You probably know that I’m a consultant and can probably guess that I have one of those “HO’s” they refer to in the “SOHO” (Small Office/Home Office) network marketing jargon. That means I get up in the morning, clear out all the under-twenties who live here, and then take the stroll down the hall to the 12x12 that functions as my office. If you live in an area where commutes are horrid and buying out of them can cost hundreds of thousands of dollars, as I do, walking down the hallway to your office sounds like a fabulous deal. Well, it is, mostly. But, as with everything, there is a trade-off.

I had pretty much settled in and forgotten my own trying experience equipping my home office until recently, when a good friend of mine decided to set up shop in his house. His daily account of his progress toward a functioning office reminded me that it really takes a significant investment of time and money to get a working office established. And you really take it for granted if you’ve always had one provided for you, as most of us do. It doesn’t seem like it would be that hard, until you set out to do it yourself.

Perhaps the fundamental difficulty is that no one wants to be an island, and when you realize you are one, it’s a little scary. I’ve seen this played out again and again as qualified professionals become overwhelmed by seemingly simple decisions.

Buying home office equipment is a good example of this phenomenon. I’ve observed many consultants’ stress levels rise when the checkbook comes out. Individuals who have had responsibility for large capital equipment budgets in previous jobs, and managed them confidently, effectively, and efficiently, suddenly, when faced with purchasing equipment for themselves, become paralyzed with indecision. They shop, talk, fret, shop some more; it’s apparent they’re agonizing over it. I think it’s economy-of-scales-in-reverse that’s the cause. For example, if you select the “wrong” copier machine for your home office, you’ll most likely have to live with it, whereas, in a corporate setting, you might just take a walk down the hallway to one of the many “better” copiers when you need to. I think the reality that you’re a self-supporting island weighs heavily at decision time.

Besides a copier, what else might you need? Well, of course, there are the computers and phone lines for yourself, computer, and fax machine. There’s hardware to support whatever type of Internet connection you choose (router, modem, etc.). I bought a plain-paper fax that also makes copies. You can’t copy an intact page of a book, but you can copy just about everything else; I really like it. Some faxes can also function as printers. I need a fairly high quality output, so I’ve had a Laserjet for years. You’ll need office furniture for your books, files, supplies, computers, and yourself. I finally bought a rack for all of the computers, but it’s still not as tidy as I’d like it to be. Ceiling trays for the wires are next on my list. You need a phone and a headset or speaker capability. Having two lines (so you can conference) is really nice. I just got a cordless phone. I’m not sure it is a feature to be able to answer the office line anywhere in the house. Remember that you’ll need to back up your systems and have a place to store the backups offsite, plus a UPS (and, for me, a generator, too).

One other thing to consider is heating and cooling. Sounds silly, but I don’t know anyone who wants to pay to heat the whole house while using only one room. The heat from the computers does the trick for me in the wintertime. I know a friend who installed a stove in his office back East. I’ve got a miniature evaporative-cooler for the summer. It’s not enough.

Another effect of the home-office-island is that you’re the single point of failure. You’re the worker and the infrastructure support. Believe me, I’ve never longed for a system administration staff more than when I’ve been alone in my office and something breaks! It seems that it always happens when you’re facing a critical deadline. At that moment, it’s hard to be the person who’s supposed to be productive and the person who has to fix the problems so that folks can continue to work. I think being “it” in this way is the single biggest drawback to working in your home.

Of course, there are some big advantages to a home office. Every day is a dress-down day for me. I’ve cut down on my water-cooler time, so I feel like I get more done. But I’m lucky because I still get to have lots of contact with others; I think the Internet really contributes to that. One plus I didn’t predict is that I get a break on my car insurance because the answer to the question “How many miles do you drive to work?” is “zero.” And I certainly like having a flexible work schedule.

All told, a home office is different, but probably not better, than the office provided in a traditional corporate setting. The real plusses come in flexibility and the minuses in the required self-support of the infrastructure.

Of course, we tend to forget the upside when we’re in the crisis phase of the self-support issue. I had to laugh as my friend relayed the latest snafu in his home office start-up effort. Apparently, he and his
spouse had agreed that she and the kids would vacate the house while he initially set up shop, because there'd be a week of upheaval while the contractor walked off an area for the office in their garage. During that time, my friend had set up temporary working quarters with his machines in the guest bedroom and his workspace on the kitchen table. Unfortunately, the contractor slipped the schedule, and his family, plus house guests, were due back that evening.

I'm still smiling as I recall the genuine panic in his voice as he vividly described the dilemma he faced. He said, "You don't understand, Tina! Half of the contents of our garage are in boxes in our front yard; I've got stuff all over the kitchen table; I've got an Ethernet cable running down the hallway; and there are power strips so dense on the floor of the guest room that you can't walk without turning something off!"

I assured him it would get better in time.

Flattening the Curve

by Hal Miller

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

I am currently (as I write this, not as you are reading it, so please don't submit!) trying to hire a sysadmin. Last time I went through this I ended up with almost nothing to choose from. Once upon a time I had 162 qualified applicants for one position. This time it was 28. Are we beginning a swing back away again from the "jobs a-plenty" market? I don't really think so yet, but it'll come.

When I had no qualified applicants, I looked at methods of educating junior sysadmins (still on top of the priority list). Maybe my thrust/aim/target would be different now? Is the need changing? Being trailblazers (sysadmin as a new profession), we have no history directly on-point to study. We can only guess as to the applicability of the histories and experiences of other professions or what combination of factors we need to consider from each of those bodies of knowledge. Might make an interesting study, but it's not my point here.

I think we need to start looking at what we might do to mitigate the problem of a cycle of too many/too few good sysadmins.

In part, I've answered the question right in the wording. We may find the curve of "available" sysadmins has much greater slopes than that of available "good" sysadmins, and a SAGE education plan can be applied to help.

However, that doesn't address the times when there may be too many "good" sysadmins. What exactly does this mean? Have we trained folks faster than the market can absorb? Are we too mobile (the economists in the audience will recognize this as "built-in transitory unemployment" due to delays in information flow, which in this silicon era is losing its viability as an economic explanation) or are we misestimating ourselves or perhaps misreading the hard-to-recognize signs of technology direction?

If it's these last ones, that's where SAGE can step in. Yes, we need to educate new folks on today's sysadmin job, plus establish some form of "continuing education" program for keeping more experienced folks current. But we also need to be anticipating new directions and getting prepared to support them, if not actually drive them. There is certainly going to be some risk involved, of putting resources into what turns out to be the "wrong" direction, but that risk is less than the risk of loss we face if we don't get on
out there in front and try to lead our industry.

So, how do we do this? For starters, we need "visionaries" for the SAGE Executive Committee (Board). This is a small group of volunteers tasked with aiming this large gaggle of overworked technology support folks in a direction most relevant for the future. That job is, though, too big already to add the task of creating and implementing new programs in keeping with the rate of technology change at the same time as the board members are working on governing the organization. There must also be some sort of Visionary Task Force, sort of an "implementer" level of the kind of thing the LISA "Advanced Topics Workshop" seems to have become. I suggest that the next board give consideration to creating such a body and empowering it as required. This calls for people to step forward, both to run for that board, and to support the elected board in carrying out this proposal.

What else can we do? Write. We have some excellent publications for floating ideas out there: slp, sage-members, etc. Put some time into solidifying the ideas you have, and write them up for one of these "journals." Don't hide behind the fear of being ridiculed -- the only bad ideas are the ones not brought to light. Don't hide behind the "I don't have time" excuse either -- none of us has the time, but we all need the benefit of those ideas to help us improve our effectiveness. When someone does bring forward an idea, feel free to step forward with constructive criticism: as I've said for over six years, the advantage of SAGE is that we all get the benefit out of the sum of the collection of small amounts of time each one of us can contribute from our daily schedules or "we're more than the sum of the parts." Together we can do those things that separately we'd never complete.

Will this flatten the curve? It should turn it into a relatively straighter line, inclined slightly upward, where the number of new "senior" jobs continues to climb while we as a profession continue to provide new value to the computing industry.

Taming the Certification Beast

by David Conran, Barb Dijker, Tim Gassaway, and Kim Trudel

Barb, Tim, and Kim are members of the SAGE Executive Committee. David is secretary of SAGE-NA.
<gassaway.barb, kim@usenix.org>,
<david.conranmember.sage-ao.org.au>

The issue of certification has been hotly debated by SAGE, on and off by the Executive Committee, since before SAGE even existed. Positions on certification have ranged from it being an absolute requisite for recognition as a "profession" to "over my dead body." Not much has changed there. However, there have been significant changes in the computing community that are causing SAGE to revisit this issue from a fresh perspective.

Seven years ago, when this debate began to rage, certification was neither an accepted nor a credible means of imparting or evaluating knowledge and skills in computing. Probably the only example of certification then was Novell Certified Network Engineer. Since then, we've seen some significant trends:

- Demand for qualified technical professionals has far outstripped the supply of ad hoc, informal, and on-the-job training methods.
- Traditional educational institutions...
have been unable to keep pace with technology and demand for skilled professionals.

- Demand for technical professionals is increasing in otherwise nontechnical organizations where employers are unable to adequately evaluate candidates.

Largely because of these trends, certification as an accepted means of identifying people at a particular skill level has burgeoned. Today there are myriad certification programs in the computing field. Just a few of these relatively new programs include:

- Microsoft Certified System Engineer. The target of much scorn, this program has evolved significantly over the years, and the current version is respected by even some SAGE members (who wish to remain anonymous).

- Cisco Certification. This program is vendor-specific, but well regarded as challenging and meaningful.

- Sun Solaris Certified System Administration. This vendor-specific program relies on course attendance rather than testing.

- Learning Tree System Administrator Certification. This vendor-generic program relies on courses, not testing.

So what? The problem such a change presents is that these certification programs, specifically, the system administration certifications, shape the public's and our employers' views of our work. Can you be replaced by someone who has completed one to four weeks of courses?

By far the most common and strongest reaction by our community to the notion of certifying sysadmins is a fatalistic fear. "The field is too diverse, the issue too broad, therefore it is impossible to do or do right." But think about that for a moment. Sysadmins are supposed to be adept at dauntlessly approaching the great unknown of problems characterized by "it doesn't work" methodically and systematically determining a cause, finding a creative solution, developing a minimal impact implementation procedure, and having a failsafe back out plan all along the way. So go to the back of the class. "Impossible" is not supposed to be in our collective vocabulary.

What about the real problem? Isn't the real problem that of proper and complete education of system administrators? Sure. But the solution to that problem first includes figuring out what knowledge and skills are required and how to evaluate whether they have been successfully imparted. Lo and behold, to solve less than 50% of the education problem, we need to do 90% of the job of developing a certification program. The two issues are intimately related. The most efficient course of action is to consider the two issues in tandem. Certainly, this is not a one-shot, over and done deal. Any decent certification program or educational guidelines will require maintenance to incorporate changes in the technology or practices.

In the last couple of years, the SAGE Executive Committee has come through a transition. Most of the current members are finishing only their first or second term. Rather than hash out the same old tired arguments, the current Executive Committee has been able to take a goal-oriented approach to old problems, including certification.

In a ground-breaking meeting in Dallas in February, the SAGE Executive Committee brainstormed on goals and strategic planning. Having decided upon some grand and lofty goals such as "become the preeminent association of system administrators," they discussed what that might entail. Not surprisingly, the beast of certification reared its indeed hideous head. With a grave determination to take one step toward either taming or slaying the menace once and for all, the Executive Committee posed the following carefully crafted question: Do we pursue certification, in some form, to further our goals? Note that determining just that question took about 30 minutes. Details on the nature of "pursue" and what form certification might take were intentionally left vague. The result was a quiet and resounding "yes."

Whew. In one afternoon, SAGE was able to finally take a step that has been seven years in the making! The next step was to add a few caveats. After a lengthy debate, the committee managed, some members a bit reluctantly, to agree to the following guiding principles for this pursuit:

- Certify people, not educational programs.

- Focus on vendor-generic "core" competency rather than a comprehensive, all-embracing specification.

- Enable expansion through specialization.

- The evaluations must be of merit, period.

- Cost should not be a barrier to achieving certification.

Believe it or not, this really was hashed out in just a few hours. The seven years of mulling and arguing had finally paid off. The beast's resounding roar was now a pitiful growl - we'd shot it with a rubber bullet. We then formed a subcommittee to determine its fate. "Great," we thought, "let's tell the members."

The designated SAGE crier diligently posts announcements of the Executive Committee's doings to the sage-members mailing list. When the post went out after that shooting, the crier was promptly riddled with many bullets - some not so rubber. The responses ranged from "how dare you" to "it's about time!"

On March 7 and 8, 1998, the Certification Subcommittee met. Our goals were

1) to address the concerns raised by the members on the mailing list and
2) to map out a top-level plan.

The Certification Subcommittee read each one of hundreds of messages in the then still raging debate on the sage-mem-
bers list. Then we categorized the concerns. There are about ten basic categories into which all the traditional concerns can be sorted. None is anything new. A letter of clarification was drafted and sent out to the SAGE-members mailing list to address all those concerns.

We then turned our attention to The Plan. It shaped up essentially like this:

- Develop a strategy for communications to/from members.
- Define a framework.
- Define skill requirements.
- Develop an implementation strategy.
- Solicit and collect bids as required for implementation.

The communications strategy was clearly the most important part of The Plan. Conducting the remainder of The Plan is wholly dependent upon the communications strategy. So the subcommittee got straight to work on it. The resulting communications strategy includes:

- a Web site to keep members informed — including a FAQ
- appointment of a diverse Advisory Council to provide constructive feedback at minor milestones
- interaction sessions where and when possible (BOFs, panels, etc.)
- periodic articles and announcements to report the status of major milestones to the membership.

So here we are. The Certification Subcommittee is pleased to announce that the communications strategy has been implemented. You’ll find the Web site at <http://www.usenix.org/sage/cet/>. The site includes the FAQ and the list of those appointed to the Advisory Council. The Plan and its consequences have been presented and discussed at BOFs at the SANS and LISA NT conferences. A certification panel was conducted at the SAGE-AU conference. Another panel will be conducted at LISA in Boston. Last but not least, David Conran, from SAGE-AU, has been added to the Certification Subcommittee.

Now we’re ready to rock. At this stage, we plan to meet with professionals in the field of certification and test development for a sanity check and to get assistance so that we don’t reinvent the wheel. Then it’s a head-first dive into program development.

As any good system administrator should have, we have a back out plan. Note that The Plan described here ends just short of actual program implementation. Once we have successfully completed The Plan, we will have:

- a defined set of skill requirements for system administrators that can be applied to education and/or certification
- a clear and concise picture of the program requirements, including costs and other liabilities

Both of these results of The Plan have inherent value even if the certification story ends there. The clear picture we get may be of a costly and ineffectual behemoth. At the completion of the planning phase, a future SAGE Executive Committee will have the unenviable task of taking that final leap to implement certification. This is by no means a given. The result of The Plan needs to measure up to our ideals of ourselves and our chosen path. In the end, the hope is that if we have to shoot to kill, the decision is made out of knowledge rather than fear.

---

**Speakers Bureau**

By Tim Gassaway and Helen Harrison

Tim and Helen are members of the SAGE Executive Committee.

<gassaway@usenix.org>, <helen@usenix.org>

The SAGE Executive Committee is pleased to announce a new online service for our members: the Speakers Bureau. This service will provide a searchable database of speakers who have said that they are available to come to your organization and speak on specific topics. The SAGE Speakers Bureau is an aid designed to facilitate local groups, conference organizers, and training managers in establishing contact with speakers of interest to your organizations. This service is provided at no cost to members or speakers.

Calling all speakers and presenters: the SAGE Speakers Bureau needs you to list yourself if you are available to give talks to SAGE members’ organizations. Your information will be on the SAGE Web pages in a searchable format that will be consistent for all speakers. All speakers have a page containing personal contact details, descriptions of their talks or training sessions, and information on when/where they are available and any associated fees. Additional information and how to register as a speaker with the bureau are available online under the SAGE pages at <http://www.usenix.org/sage/>.

If you are not a speaker but have heard a good talk that you think would be of interest to SAGE members, please let the speaker know about this service or pass contact information along to <sage-auc@usenix.org> so that we can suggest that the speaker list himself with in the Speakers Bureau.

Obligatory disclaimer: the SAGE Speakers Bureau is a repository of speaker-supplied contact information. SAGE makes no claims as to the accuracy of the information.
Sometimes we change our minds. Sometimes external forces change and cause us to retract things said in the past and say the opposite. This is one of those times.

At many a LISA conference I’ve heard people complain about their horrible work environments. “My management doesn’t appreciate sysadmins!” “They won’t budget for the right things!” “I can’t convince anyone that a written security policy is better than random hole tightening!” “Management never goes to bat for me, but won’t take responsibility for decisions they force on me!” or “My management wouldn’t endorse my plan to install virus checkers, but a year later they’ve all lost their files they’ve come up with an idea to fix it: install virus checkers.”

My response has always been serious and responsible: Sysadmins should become advocates for change. “Be the innovator, the force of change that educates management, improves the network, and makes it all happen!” I’ve said this a million times in those impromptu chat sessions at USENIX/LISA conferences. I’ve said it on the Net. Heck, I’ve written a paper (see the LISA’98 program) on some techniques to help break the cycle while maintaining your sanity.

No more. My new advice is much more simple: please quit. That’s right. Please quit. If you’ve tried your best to resolve problems, then I’m forced to recommend the final solution: please quit.

Statistics show that today sysadmins are in incredibly high demand. I’m told that there are 20 jobs for every person job hunting today. If this is true, then why stay in a bad situation? Jump ship! Find a company that appreciates sysadmins, and sign up with them. It is silly to stay in a bad situation when there are so many companies around offering good situations.

You say you’re afraid the company will fall apart without you. What drugs are you on? Do you really think your involvement makes a big enough difference to put them out of business if you leave? Sure, your domain of responsibility will suffer for a while, but you’ll be replaced by some other poor sucker. You are the size of a fly compared to the dead horse of your company. So beat it!

If you leave, it is their problem. Maybe they’ll be just fine without you, and you’ll be happy in your new job. A win-win situation. However, maybe a couple months after you leave their held-together-with-spit-and-glue network will crumble, and they will realize that they need to start being one of those companies that takes information technology seriously. You’ll never see the positive results of your leaving, but the company will be better for it. Or maybe it will go out of business. Is that your fault? Not likely. That’s the responsibility of the board of directors.

In today’s new business environment, companies that manage information better than the competition will win. Those that can’t learn how to manage information will lose (“go bankrupt” or “be bought out”). Sysadmins manage the systems that manage information; they are the information management experts. A company that treats sysadmins like second-class citizens, or that doesn’t treat them as part of the business process, is demonstrating that its management doesn’t “get it.”

In the 1970s, American industry was nearly wiped out because it refused to modernize its manufacturing capability. Ask my father, who worked in the metals industry and saw an outright refusal to modernize practices until company after company paid the ultimate price. As we move to an information economy, certain companies will again refuse to read the writing on the wall and be wiped out.

Barrons, The New York Times, The Wall Street Journal, and Fortune used to just warn about the new demands of the information economy. Now these demands are an accepted fact. If your company’s board of directors lives under a rock and refuses to hear the warnings, it isn’t your fault.

In reality, we are strengthening the economy. Moving better sysadmins to the better companies and letting Darwinian capitalism run its course results in companies that are better adapted to the new economy. The sad, cold truth about capitalism is not that the strong survive, but that the weak die.

Do you really want to add to the problem by extending the life of your company by months, if not years? The patient is already dead; its heart is still beating because the brain is too damaged to know to stop it from beating. Do not resuscitate.

Sysadmins are in demand today, but how long will that last? Certainly five years from now supply will catch up with demand, and we won’t be able to be so willy-nilly about changing jobs. My advice has an expiration date that is unknown, but very real. It may be a good idea to use these years to jockey for a position that will be good once times get hard.

I don’t recommend you quit right away, of course. First make an effort to change things: a real sit-down-with-your-boss-and-talk effort, a real if-at-first-you-don’t-succeed-try-try-again effort. I guess that’s always been my advice. I just ended before I got to the final paragraph.

If you’ve made an honest effort or two or three and your company doesn’t offer you a compelling, interesting, and empowering work environment, it’s time to quit. I would never recommend someone throw good money after bad; I would never recommend that someone give himself a heart attack from the stress of trying to change a company on a collision course with destiny. Nobody knows how long sysadmins will remain in such incredible demand. Wait too long, and we won’t be in such an excellent position. Until that happens, just quit!
Effective Perl Programming: Simplicity Is a Good Object

In my previous column, I discussed Perl's framework for object-oriented programming. In this column, I discuss how some common object-oriented programming paradigms can be addressed with Perl. As before, I'm assuming you have a Perl reference handy to look up any features that you haven't already encountered.

Member Variables

By and large, objects in Perl are implemented using blessed hashes. Used this way, Perl hashes are analogous to "structs" or "records" in other programming languages. By extension of the analogy, each key-value pair in a hash represents a member variable. In reality, there are considerable differences in implementation: C++ structs/objects have a fixed number and type of members and a fixed layout in memory, while Perl hashes have an unconstrained composition and size. However, from a user's standpoint, things seem pretty much the same:

// C++ version:
joseph = new Person("Joseph", "Hall");
first_name = joseph->first;

# Perl version:
$joseph = new Person('first' => 'Joseph', 'last' => 'Hall');
$first_name = $joseph->{first};

The analogy works up to the point where you introduce the notion of public versus private member variables. Many object-oriented programming languages support a partitioning of the namespace of member variables such that some member variables, the "private" ones, are visible only within methods belonging to that object's class, or perhaps to methods derived from that class. Put another way, the private member data are hidden. But "public" member variables are accessible everywhere. Although Perl has some namespace-related features, like packages, Perl does not have any features that directly address object data hiding. So long as Perl objects are implemented as hashes, member variables (which are really just key-value pairs, remember) are always visible to any code using those objects. This becomes a fundamental rule in Perl, at least as the language now stands: in Perl, all member variables are public.

You can struggle against this rule, and you can do various things to try to work around it (and I'll even show you one later), but you can't alter Perl's fundamental nature – not by writing Perl code, anyway. In a bit, I'll discuss ways of dealing with the lack of data-hiding features. First, though, let's turn away from member variables and look at class variables.

Public and Private Class Variables

In the lingo of object-oriented programming, a "class variable" is a variable whose single instance is shared by all members of that class. In Perl, classes are packages, so you might think that a Perl package variable would be the natural representation of a class variable in Perl. If so, good thinking! For example, suppose we want to extend our Person class from the last article to include a count of all the Person objects created so far. We need only add a package variable and modify the constructor accordingly:
package Person;
sub new {
    my $class = shift;
    my $self = { @_ }; # constructor for class Person
    $count++;
    bless $self, $class;
    # $Person::count contains the count
    # return a new Person
}

Within package Person, $count contains the number of times the Person constructor has been called. Outside Person, we can still access the value with the qualified name $Person::count. We have created a public class variable. Suppose, however, that we would like to make $count private – in other words, visible only to methods of the Person class. Once again, Perl lacks features allowing us to precisely address this data-hiding need, but in this case, we can come pretty darn close to what we want by using another feature. So long as we can group the methods of class Person into a single file or portion of a file, we can use a my variable to hide $count from the rest of the world:

    
    package Person;
    my $count; # $count is visible only within the braces
    sub new {
        my $class = shift;
        my $self = { @_ }; # put all Person methods inside these braces
        $count++;
        bless $self, $class;
    }
    sub get_count { # add a class method to return the value of $count
        my $class = shift; # should be 'Person' or subclass; we ignore it anyway
        $count; # return value of $count to the outside world
    }
    # $count no longer visible beyond here
    # some intervening code ...
    print "the current count is ", Person->get_count, "\n";
    # example usage

A more conventional and advisable practice is to write Person as a module (modules are, alas, a topic for another day). Then all of Person goes into a single file, which provides a scope for the my variables and eliminates the need for the enclosing braces. Sometimes you may want the braces anyway, but as part of a BEGIN block:

    BEGIN { # all code in here is executed at compile time
        package Person;
        my $count = 1000; # initialize $count to 1000 before new is called
        sub new {
            # rest same as above; methods work the same
            # enclosed in BEGIN
        }
    }

Putting my $count = 1000 inside a BEGIN block ensures that $count is initialized before any of the methods of Person are called by the outside world, even if (for example) the code for Person appears textually after the first code that calls Person::new. The same thing is happening if Person is implemented as a separate module incorporated with the use directive – the code for Person is included as if it were all surrounded by a BEGIN block.
Private Member Variables, Sort of

Okay, so Perl doesn’t really support private member variables. But no doubt if you’ve been using Perl for long, you’ve seen all kinds of weird language gymnastics that make seemingly impossible things possible. What about this case? Is there something we can do to implement private member variables in Perl? My answer is still no, not really. There just isn’t a way to hide private member variables efficiently in Perl. This doesn’t stop you from taking stabs at it. For example, the perltoot man page suggests using a closure (yet another topic for another day) as an object. True, data can be hidden very thoroughly within a closure, but (1) closures make very bulky, space-inefficient objects and (2) you run into a Gödel-Escher-Bachish mess trying to come up with a way to reveal the contents of the closure to class methods without once again revealing them to the rest of the world. The perltoc example is not well written, and in any event the underlying concept is flawed.

If you have a limited amount of private data for each object, you might try something along the following lines. I’ve written some code that associates a unique identifier with each Person object:

```perl
package Person;
my $count;
my @PRIVATE_ID;  # also visible only within the braces

sub new {  # for the my variables (or use a separate file)
    my $class = shift;
    my $self = bless { @_ }, $class;  # bless before hash assignment
    $count++;  # debug before hash assignment
    @PRIVATE_ID{$self} = $count;
    $self;
}

sub get_id {  # should be reference to Person object
    my $self = shift;
    $PRIVATE_ID{$self};  # look up id for this object
}

sub DESTROY {  # destructor cleans up as necessary
    my $self = shift;
    delete @PRIVATE_ID{$self};
}
}
```

Here I’ve used an object reference, $self, as an index into a hash, %PRIVATE_HASH. The hash is shared by the methods of Person and is inaccessible elsewhere. After a few moments of study, the way this code works may seem clear to you, but be careful. The code %PRIVATE_ID{$self} probably doesn’t work like you think it does. Perl cannot use a reference directly as a hash key because hash keys must be strings. The reference $self is converted to a string by Perl; the string will look something like “Person=HASH(0x18d76b4)”. This string is guaranteed to remain unique so long as the object it refers to hasn’t been destroyed. It’s not a bad idea to employ a destructor (the subroutine named DESTROY, which is automatically called when objects of class Person are destroyed by Perl) to ensure that the contents of the hash remain consistent with the objects currently in existence.

This technique is speedy enough for many uses and is certainly space-efficient. Is it a hack nonetheless? I’d have to say yes, it is.

Keep It Simple, Stupid

The simplest and most efficient way to implement private instance variables is do it by
The one thing I don’t recommend is creating complex boilerplate or conventions to impose your object-oriented desires on Perl. Perl has a simple object-oriented framework, and it is best enjoyed and appreciated for what it is.

convention. Give private variables special names: for example, begin them with an underscore. Or, perhaps, provide a method-based interface for manipulating instance variables:

```perl
package Person;

sub get_first {  # ... stuff from above ...
    my $self = shift;
    $self->{first}
}  # return value of first

sub get_last {  # shorter version
    shift->{last}
}

Or even:

```perl
sub first {  # function as a means of setting as well as retrieving an object value:
    my $self = shift;
    $self->{first} = shift if @_;  # $joseph->first to get $joseph’s first name, and
    $self->{first}('Joe') to change it.
}

The latter version permits a single method to function as a means of setting as well as retrieving an object value: $joseph->first to get $joseph’s first name, and $joseph->first('Joe') to change it.

The one thing I don’t recommend is creating complex boilerplate or conventions to impose your object-oriented desires on Perl. Perl has a simple object-oriented framework, and it is best enjoyed and appreciated for what it is. Perl doesn’t make a very good Smalltalk or Eiffel or (especially) C++. That works both ways, though. Those languages make lousy Perls, and I’ll take simplicity over complexity every time I can get away with it. By the way, I ran out of space for my CPAN subclassing example, but you can expect to see it in a forthcoming column.
interview with
L. Peter Deutsch

Stig: Hi, Peter. You have been working on free software for quite a long time. How long have you been doing it, and what influenced you to work in that direction?

Peter: I’ve been working actively on free software – more precisely, what is now called freely redistributable or open-source software – for about 10 years. The only substantial free software program that I have done is Ghostscript, and I started work on it almost exactly 11 years ago.

What motivated me to do that in the first place was really two things. Sort of the proximate one, I guess, was having known Richard Stallman for quite a long time and liking the idea of the GNU Project while at the same time recognizing that it was somewhat quixotic and that, in my opinion, you couldn’t make the whole industry run on free software. But I felt that having free software – at least free system software – was a really good idea.

But I think the more important motivation for me was the fact that I basically grew up professionally in the 1960s. It was a world in which the commercial software market, as we understand it today, basically didn’t exist. The interesting things that were being done in the 1960s were all being done by – you should pardon the expression – kids. People in their 20s, maybe early 30s, who were doing it because it was fun and because it fed their egos and because it was a contribution.

You’ve probably read Steven Levy’s Hackers. And one of the things that impressed me about Hackers was that in fact he did do quite a good job of capturing what I remember as the spirit around the Tech Model Railroad Club (one seminal group of people who were developing that kind of technology in that kind of spirit in the 1960s).

So that attitude toward software – namely that it’s done by people who care about it and the way that you move things forward is by trading and cooperating – stayed with me as what seemed to me like the best way to move software technology forward. And that, in turn, gave the GNU Project natural appeal for me. The reason that I started working on free software, at the particular time that I did, was that I had gotten very disenchantened with working at Xerox PARC over the preceding few years.

I was at PARC from 1971 to 1986. For the last five years of that period, I was working with the Smalltalk group, and we were getting pretty frustrated at Xerox’s apparent inability to turn Smalltalk into any kind of product or to just release it to the world. And so I did some looking around outside Xerox in 1983 and 1984. Then, in early 1986, I decided that I’d had it with Xerox and that I wanted to get on with my life. So, I arranged for a year’s leave from Xerox, starting in July of 1986, and I went to work with another startup for a few months.

My plan was to just take the remainder of my nine months off and maybe learn some new languages, write some music, and do other nontechnical things. But I had made reservations to go to the first OOPSLA up in Portland. So I went up there, and of course I saw my old buddies from Xerox, and Adele Goldberg said, “With or without Xerox’s blessing, we are going to start a company and do Smalltalk.” And that was the one opportunity that I was willing to let me change my plans for a leave.

But at the same time, for some reason, I knew that, because this was a time when I was making a shift in what I was doing professionally, this was a really good time to do some serious free software project as well.

Stig: Why Ghostscript?
Peter: I traded a couple of emails with Stallman. First, I proposed doing an incremental linker, and he said “No, we’ve already got somebody doing that. How about a free window manager for X?” I said, “No, I’m not interested in doing that. How about PostScript, because it combines language technology, which is something that I know a lot about, with 2-D graphics, which is something I only know a little bit about, but would like to learn more about?” So that’s how a PostScript interpreter got chosen as the project that I was going to do.

Stig: That’s interesting. I think that the general assumption is that most people who write free software are motivated strictly by a strong interest in the application(s) that they work on. You actually shopped around for a project before you began work on Ghostscript, and you were guided, in part, by Stallman?

Peter: Well, I wanted to do a free software project and there was this little bit of negotiation about which project it would be. I think I would have been just as happy to do an incremental linker or possibly several other things. And, you know, people asked me why I was doing it, and I gave them the following answers:

- I like the GNU free software enterprise, and I want to contribute something to it.
- It’s an opportunity for me to explore technical challenges in an area that I don’t know all that well, although it leverages what I do know.
- It’s an opportunity for me to learn something, learn hopefully quite a bit in a domain that I don’t know much about, namely, raster graphics.
- And it just sort of looks like it could be fun.

Also, something that I already knew in 1986 was that some day I wanted to have my own software business. I had seen the internal process at Xerox being really screwed up and — well, I had seen the internal process being screwed up at every company I had ever worked for.

Stig: By bureaucracy?

Peter: A combination of things. I worked for Adele at ParcPlace for five years, and I was chief scientist there. I was one of the three or four key people that got their product working. Even at the end of those five years, I started to see process dysfunction at ParcPlace as well.

At that point, my intention was to go off to do the commercial Ghostscript business full time.

Stig: So, for the first five years that you worked on Ghostscript, you were also...

Peter: I was also working at ParcPlace. My hire letter at ParcPlace explicitly said that I was allowed to work on Ghostscript on my own time and on my own equipment and ParcPlace had no claim on it.

Stig: Ghostscript started out as an evening project?

Peter: Yes, that’s right. It was actually closer to the first six and a half years that it was an evening project. From ParcPlace, I actually went to a full-time job at Sun for about a year and a half.

Stig: What was the personal cost?

Peter: It didn’t seem like that much of a sacrifice, but it meant that I didn’t have much of a social life. At that time, it didn’t matter very much to me.
Stig: In hindsight, would you have liked to have had more of a social life? Or are you entirely happy with the way that things have worked out?

Peter: You know, I tended to be a fairly solitary person. I have been in a relationship now for pushing ten years, but my partner and I don’t even live together and that seems to be the way it works best for me.

So no, I can’t say that during very much of that time I was feeling there was a personal cost associated with what I was doing.

Stig: Did you flirt with burnout during any of those ten years?

Peter: I can remember being close to burnout three times. One was at Sun, where I was burning out not because I was working too hard but because I felt like I was flailing around in a vacuum. Sun’s dysfunction, at least from my point of view, was not that there was too much bureaucracy, but that they didn’t give me any support. That is in terms of person to person; the equipment and the money at Sun were fine.

The other two times were both since I’ve been doing the Ghostscript commercial business full time. They were both basically caused by the business growing to the point where I could no longer do the things that had to be done and it took me just a little bit too long to recognize that I had to hire somebody to do them. During the process of creating Ghostscript, I don’t remember ever feeling that I was flirting with burnout.

Stig: Why did you originally choose to license Ghostscript with the GNU General Public License (GPL)? Was it because of Stallman’s influence?

Peter: Well, I believed in the GNU Project then, and I still do today. I think the GNU Project is a fine idea. And the thing that appeals to me about the GPL is the notion of carving out an area where software will remain free.

Stig: Did you carefully analyze the GPL before you chose to use it?

Peter: I did not. At the time, 1986, there weren’t a lot of alternatives. And, in fact, there’s something important that I was about to say and this is a good time to say. When I started working on Ghostscript, I said to Stallman, “I want to keep the copyright on this code because someday I may want to license it commercially. I will release it with the GNU license, but I won’t transfer the copyright. I’ll keep the right to license it in other ways if I want to.” And Stallman was not happy about this, but he basically said, “We would rather have it this way than not have it.”

And I did something which in retrospect I wish I hadn’t done. Which was that I promised Stallman in writing that all future versions of Ghostscript would be released with the GPL. And now I sort of regret that. I don’t feel I can go back on my word, and I’ve done something which I’m sure Stallman doesn’t like, which dilutes that a little bit. I was very unsophisticated about free software at that time.

Stig: Unsophisticated how? About licensing? About intellectual property?

Peter: With respect to licensing, I didn’t realize there were other options for drawing the lines about permitted commercial uses, and I didn’t appreciate the existence of the GPL’s “free rider” problem.

I also was not as experienced as I am now about intellectual property. I had run into patent protection issues when I was at Xerox and I thought that a good deal of those issues were silly. At the same time, I understood the value of code copyright. I guess it’s harder for me to give up the notion of code copyright being valuable than the notion of patenting being valuable, but there are wild men like John Perry Barlow who don’t even really believe in copyright.

L. Peter Deutsch

"That attitude toward software – namely that it’s done by people who care about it and the way that you move things forward is by trading and cooperating – stayed with me as what seemed to me like the best way to move software technology forward."

<ghost@aladdin.com>
<http://www.ghostscript.com/>
<http://www.cs.wisc.edu/~ghost/index.html>
I guess the position that I’ve come to take is that I find it easier to justify protection of the copying of artifacts than protection of the copying of ideas. But at the time I began work on Ghostscript, I hadn’t really thought about these issues in any concerted way.

Stig: I tend to unfavorably describe intellectual property as a “legislative distortion of reality” because it tries to make ideas and their expressions “ownable.” Is it fair to say that intellectual property is a reality distortion, and do you think it’s justified?

Peter: I guess there are many realities. The reality that justifies what is called intellectual property protection is that creators should be rewarded for their creations. I think that argument has a real basis.

I think that certainly the current patent system and some aspects of the current copyright system are damn poor ways of doing it, but I think that the desire to do that is legitimate.

Stig: I agree. Still, do we have the right social contract for intellectual property? Let me rephrase that last question:

The reality is that there is no incremental cost for making a copy of software or of anything else that is essentially just information. And so we have created a legal fiction that artificially increases the cost of making copies, that creates economics of scarcity where we might otherwise have abundance.

And this is one possible incarnation of a social contract or a pact between authors and consumers that says, in essence, “you did this work, we didn’t pay you while you were doing it and we are not going to stifle you as we use the fruits of your labor.” What kind of fallout is there from the way that our society has chosen to reward authors? This is an overly broad question.

Peter: No, it’s a very philosophical question.

Stig: There’s a fine line between a philosophy and a business strategy.

Peter: Yes! Although it’s a pretty blurry line these days.

I guess I would answer the question the following way: there is a fine line between discovery and invention. In my opinion, that line is fuzzy enough that it doesn’t seem to be appropriate to have a social contract which says that invention should be owned. I can’t justify, on any basis of principle, providing monopoly protection on something that is discovered and that has no cost of replication.

Stig: This is patent law?

Peter: Right, this is patent law. And that, basically, is the philosophical basis on which I oppose patents, certainly for software. And maybe for many other things as well but that’s a discussion I don’t want to get into right now.

My position on patents, I think, is fairly well thought out and I am prepared to defend it. My position on copyrights is self-contradictory. I don’t quite know where it’s going to lead.

With respect to copyright, the thing is that constructed artifacts have a larger cost of creation. a larger cost of creating the artifact in the first place.

Stig: What is a constructed artifact?

Peter: A program is a constructed artifact. It’s not something that you discover, and it’s not something that you invent. You can invent a way of doing something, but in order to reduce it to practice, there is quite a lot of work involved.

Stig: Yes, implementation.

Peter: Design and implementation. Design sort of straddles implementation and invention. So, if we are going to imagine changing the social contract about software so that software is considered to be a social entity rather than a privately owned entity, there is still the question of how you pay for its creation because its creation isn’t cheap.

And that’s the reason that I’m now straddling the fence on software copyright. That’s also the reason why I created the Aladdin Free Public License (AFPL). I’d like to talk for a bit about licensing and what one hopes to accomplish with licensing. And then we can talk in more general terms about cooperative software processes.

I’m going to compare the the GNU license and the Aladdin license in a way that is a little unfair to the GNU license. The GPL takes the point of view that it rewards cooperation by making the work done cooperatively available freely to anyone who is willing to play by those rules, but it does not draw a hard line that prevents that work from also being used in a way that makes money for other people who weren’t involved in its creation.

Stig: I’d like to point out that I think “cooperative” is the key word here. I prefer to describe software that is freely distributed in source form over the net as cooperative software and not as free software. I really like to make a distinction between free software and cooperative software.

Peter: Yeah, yeah. Cooperative software, I think, is a much better word because I think that it’s the essence of what is valuable about the GPL. That’s right.

Stig: Though the manifesto portion of the GPL stresses that “free” refers to freedom, the terms of the GPL also lean heavily in the direction of gratis redistribution. Consequently, I think of free software as the software that has either been donated — as a gift, as a write-off, or as a loss leader — or that has been somehow “paid for in full” and transferred to the public sphere.
Cooperative software, in contrast, is all of the software that is being developed in the particularly open and non-proprietary way that is now becoming common on the Internet. Once cooperatively developed software (or even proprietary software) has been paid for, then it can (or should) become “free.”

**Peter:** You know, I think this is very much in the spirit of the original intent of the US copyright system. The article in the Constitution that authorizes a copyright system says the system should grant a monopoly only for a limited time and specifically for the purpose of encouraging progress in the useful arts, or something like that. By implication, copyrightable things would naturally be in the public domain if it weren’t for this trade-off.

One interesting way to slice the issue of getting software paid for is payment in advance versus payment after it’s done. For payment in advance, I think both providers and buyers in the commercial world today don’t have much trouble accepting the idea of “paid in full.” It’s payment after completion that brings up all the controversy about what kind of charging is legitimate. So it seems to me that the way to get software to be created more cooperatively, and to get it to be paid off more readily, is to find mechanisms to get it paid for in advance. Maybe just better mechanisms for getting groups of advance funders together with authors would be enough.

That would still leave the problem of taking responsibility for the software after it’s written. This is a big problem with free software that outfits like Cygnus only partly solve. I think the GNU people sort of understand the issue about paying in advance versus paying afterwards, but I also think the way they present it turns people off.

**Stig:** The GPL doesn’t address the issue of making money for people who create and maintain GPLed works. It’s just that, de facto, if you hold the copyright, then you don’t have to use the GPL, and that’s what you’ve done with Ghostscript.

**Peter:** That’s correct. And as far as I know, I am the first person, and so far perhaps the only substantial person, who has taken advantage of that. As you recall, I promised Stallman that I would continue to distribute Ghostscript with the GNU license. But I saw a number of companies bundling Ghostscript with commercial products without just barely complying with the letter of the GNU license, so I decided that I did not want to make Ghostscript as available for commercial distribution as it would be with the GNU license.

And so I am now continuing to distribute Ghostscript with the GNU license, but about two revisions back from the version with the Aladdin license. The latest version is now always called Aladdin Ghostscript instead of GNU Ghostscript and is released with the Aladdin Free Public License.

**Stig:** I gather that some people, perhaps even many people, are disappointed by your decision to stop using the GPL for all versions of Ghostscript.

**Peter:** Then perhaps the act is not properly understood. I put a lot of thought into what I saw as the flaw in the GNU license when formulating the Aladdin license. The essence of the Aladdin license I can describe in one sentence, and it is very much about social contracts.

Namely, if you are willing to play by what I think are the 1960s rules, then the Aladdin license gives you exactly the same rights and benefits as the GPL: it’s free to use, it’s free to copy, and you are free to modify it. All of those things. In a nutshell, I see the 1960s rules, or the cooperative rules, this way: “everybody contributes, so everybody benefits.”

Unlike the GPL, I make a very solid distinction between distribution as part of a commercial endeavor and distribution not as part of a commercial endeavor. Distribution not as part of a commercial endeavor is covered by essentially the GPL rules, while distribution in any commercial endeavor is not permitted by the Aladdin free license.

The philosophical weight of this is that if you want to play by cooperative rules, you get the benefits of Aladdin’s work within the context of those rules. If you are not playing by the cooperative rules, then it’s going to cost you something to have the rights to get the value from Aladdin software.

**Stig:** The name of your license changed. It used to be the Aladdin Ghostscript Free Public License, right?

**Peter:** Right. It’s now simply called the Aladdin Free Public License because a couple of other people have used it and I rewrote it to take out specific references to Ghostscript.

**Stig:** Who else uses it?

**Peter:** I think Russ Nelson at Crywrr was speaking of using it for packet drivers for network controllers: software designed to interface to specific hardware. A few other people have asked me for copies, but I don’t remember who they are now.

**Stig:** Okay. Well, as I understand the Aladdin license, permission is granted to include Ghostscript in its original or derivative form on any media for sale that doesn’t include any software whose noncommercial redistribution is restricted, such as commercial software, right?

**Peter:** Correct.

**Stig:** Then the Red Hat box set cannot include Aladdin Ghostscript (the 5.x versions) under the terms of the AFPL because the distribution medium also contains licensed proprietary software: Metro-X, BRU2000, and the RealAudio player and server.
But Red Hat could distribute Aladdin Ghostscript on their Power Tools CD-ROM without violating the AFPL terms if everything else packaged together with Ghostscript is “redistributable for noncommercial purposes without charge.”

**Peter**: That’s right. Red Hat could also offer to throw some money at me to license Aladdin Ghostscript for their box set, and I probably wouldn’t charge them very much because most of the stuff that they distribute is free and because they are providing value to people who are working in the cooperative arena.

**Stig**: How much are you personally involved with the licensing of Ghostscript? The Ghostscript Web page directs licensing inquiries to Artifex Software.

**Peter**: I set up Artifex as a separate entity so that I could be as free as possible from the day-to-day operations of the business. At first, they were simply a licensing agent for me. Now they provide support and are developing (with my cooperation) some additional software products, some of which will not be distributed with the AFPL or GPL. They recently hired a really excellent VP of engineering, so it won’t be long until I reach my goal of being able to walk away from the Ghostscript business if I want to.

Miles Jones, president of Artifex, and I got to know each other over a period of many years before I hired him, and he understands pretty well my feelings about cooperative software. He understands the value that he is getting from continuing to release Ghostscript with cooperative licenses. In particular, he understands that the noncommercial releases benefit Artifex by bringing in an incredible amount of free testing and bug reporting. In some cases, they even get (through Aladdin) significant additions or improvements to the code. By significant, I mean more than just a few lines of code. If, for example, somebody sends me a new driver, what they typically do is they just copy the copyright notice from the existing drivers, which says “copyright Aladdin Enterprises.”

For contributors where I think there might be an issue, I explain to them that they are welcome to leave their own copyright on their work or to put an FSF copyright on it, but if their distribution terms are more restrictive than the Aladdin distribution terms, then I’m not willing to distribute their software because I just don’t want to do the bookkeeping. In that case, I am still willing to put the reference to their software in the Aladdin documentation.

**Stig**: That makes the barrier to using those enhancements so great that most people won’t ever find them.

**Peter**: Well, actually, I put references to them in the second file that any new Ghostscript user would read (after the README). I’m happy to put those references there. And as it’s turned out, the only such things that I can’t distribute are a few device drivers.

But generally I’ve said, “I would like you to transfer the copyright to Aladdin. I promise that your code will continue to be distributed under the Aladdin license, so you and everyone else will continue to have access to it in source form. But by doing this, you do give me the right to license it commercially.” So far, nobody has objected.

**Stig**: This reminds me of the way that the Free Software Foundation (FSF) requires a copyright assignment for all contributions to FSF-maintained packages, like Emacs or GCC.

[Post-interview note: Stallman’s rationale for this is that the FSF may be unable to enforce the GPL in court if the ownership of the copyright is in question or if a copyright holder doesn’t have the time or resources to enforce the GPL license terms. Personally, I would prefer to see FSF work toward a recognition of the copyleft concept within the court system. Anyone should be allowed to demand adherence to a copyleft-style license when a breach is detected.]

**Peter**: Now, I don’t require a copyright assignment, but it does make things simpler. Anything that has a GPL on it I am willing to distribute with GNU Ghostscript. I don’t care whose copyright it is.

**Stig**: There are two branches of Emacs development, and I used to work on the non-FSF branch, XEmacs. A lot of us – I’ve spent a lot of time working on XEmacs – rejected the notion that we ought to assign our copyrights to the FSF; we felt that releasing our work licensed with the GPL should have been sufficient.

There are technical reasons and personality reasons for the split between the two camps of Emacs developers, but an unwillingness to assign copyrights to FSF has definitely contributed to the divergence. I’m glad to see that you take the XEmacs-camp mentality of trusting the licenses of other developers at face value.

**Peter**: Yes, for distribution with Aladdin Ghostscript, I am willing to distribute anything whose redistribution terms are at least as liberal as those of the Aladdin license. I do take the licenses applied by other developers seriously, just as I do my own.

But taking licenses seriously means being prepared to defend them when necessary. One of the benefits of having the goodwill of an enormous user community is that people tell me when they see what appears to be copyright abuse – either Aladdin or GNU Ghostscript in a commercial product, usually a CD-ROM in the back of a book. I’ve enforced the copyrights on both versions by writing to publishers, getting them to change what they publish, and in some cases getting retroactive royalties as well.

**Stig**: It’s interesting to note that AFPLed code and GPLed code are immiscible. This is because your license terms are slightly more restrictive than those of the GPL.
Peter: Yes, my terms are more restrictive than those of the GPL. I can’t distribute GPLed software as a part of Aladdin Ghostscript because the GPL forbids it to be distributed with anything that has a more restrictive license.

[Post-interview note: Linking Aladdin Ghostscript against libraries covered under the library version of the GPL (LGPL) would not be a problem, though.]

Stig: This means, then, that Aladdin Ghostscript and the GV front end to Ghostscript cannot be integrated into a single application.

Peter: That’s right. The GPL says that they can be “aggregated on a storage medium,” but they cannot be commingled.

I don’t think that the distinction drawn by the GPL is a very defensible one. When you start talking about APIs and dynamically loadable libraries, then I think the line between when something is “aggregated” and when it becomes a part of another program becomes quite blurry. I don’t want to have a similar exception in the Aladdin license simply because I don’t think that it can be defended in court.

But aggregation and commingling are not an issue with software that has very liberal redistribution terms. Such code can be readily incorporated into either GPLed code or into Aladdin Ghostscript. The IJG JPEG library, zlib, and the PNG library are examples.

Stig: Let’s get back to the state of cooperative software in general.

Peter: We agree that when people are able to work and contribute cooperatively to the evolution of the piece of software, I think everybody benefits. To me, the difficult question is basically, how are the people going to get paid?

The FSF answer is that the way to do this is by providing services and I don’t . . .

Stig: But the initial authorship of the software is a service. It’s work, and it is of value, but the FSF seems to expect that the world’s software developers ought to write software for free and then make their money some other way.

Peter: Right. And that is a part of the FSF argument that I just don’t buy.

But there is an aspect of the FSF argument that, in fact, responds to what you just said. Namely, is there enough of the right kind of demand that people could be paid for writing software and basically paid once for their work?

Stig: Sort of like working as a contractor, but paid after the fact.

Peter: Right. At this point a noticeable, though perhaps not significant, part of my income comes from doing paid enhancements. I make it very clear up front that I will do paid enhance-

ments to Ghostscript only if I own the copyright and I get to distribute the enhancements on exactly the same terms as the rest of Ghostscript, including free distribution. So far, nobody has objected to those terms.

Stig: That’s very different from the terms that most people have in their employment contracts, where everything they do is deemed to be the intellectual property of the employer.

Peter: Yes, that’s right. I have lots of leverage when I work to enhance Ghostscript, particularly since most of that work is paid for by those who are already licensing Ghostscript commercially.

Stig: Okay. We’ve been talking about licensing, philosophy, copyright, and social contracts for a while. Let’s talk about the status of the cooperative software world right now. I want to get your impressions of who is doing the work and what they are getting out of it.

Peter: I am sort of out on a spur of the free software world. I am not really in the middle of it.

Stig: Everybody uses Ghostscript.

Peter: Yes, but I’m still not really very much a part of the world of cooperative software. As I said earlier, I am a somewhat solitary person. So, for example, I didn’t know about the Apache Group’s development process, which you pointed out to me. I knew that Apache was the leading HTTP server, but I didn’t realize that it had such a fascinating social process behind its evolution. I think it’s fabulous.

So I can only give you, in general terms, my impressions of the people that I know who have contributed various bits of free or free-ish software. Those people have not seemed to be motivated by a grand social goal. Like me, they simply seem to feel that the people whom they know – the software development community, and to some degree the software user community – are better off if software is done this way. They feel that better software gets produced in a more timely way, and it becomes more available to people.

I also think that some people contribute free software to enhance their professional reputations and perhaps to further their careers in the profit-seeking world.

Stig: Okay. What’s your impression of the demographics of the cooperative software developer community? I’ll give you my impression first.

From my time spent working on XEmacs and watching the mailing lists of other freely redistributable software projects, I’ve come up with a sense of the demographics of who hacks on cooperative software. My numbers aren’t well-documented, so let me bounce them off of you.
I think that about 60% of the hacking, beta-testing, and bug-hunting in the freely redistributable software world comes from university environments and is done mostly by undergrads and grad students.

Probably about 25% of the hacking gets done by people who are in workaholic mode. They have another computer job, and hacking free software isn’t it. They may have less of a discernible “life” because their hobby is so closely aligned with their day job. This group may generate more useful lines of code because of greater experience, or it may generate fewer because these people have (theoretically) less free time.

Next, I would say that about 10% work at companies that have chosen to use and support freely redistributable software in some peripheral way. They have other job responsibilities, but they are responsible for supporting a tool because it’s used internally. Their work will often result in improvements which become available on the Internet or may result in supporting other users of the same tool at other companies.

And then the final 5% of the people who actually work on freely redistributable software are supported entirely by the budding cooperative software industry. This would include most of the people working for companies like Cygnus Support and Red Hat Software.

What’s your take on this?

**Peter:** I have a slightly different take, but it’s because I slice the world a little differently. I think in terms of people who are generating lines of code.

Your assessment may be right in terms of time spent, but I think that if you ask the same question about people who are evolving and maintaining and shepherding the free software that has stood the test of time and that is of sufficiently high quality that people actually use it, then the breakdown will be substantially different.

Partly it’s because students eventually graduate. If their software continues to be out there, then it’s quite likely that they will continue to be heavily involved with it, even as they move on to something else as a primary commitment.

**Stig:** Only if they have the time...

**Peter:** Well, yes, if they have time.

Also, I think that your estimate of the proportion of people who are being supported by free software is probably lower than 5%, and I don’t have any theory on what proportion of the non-University people are classifiable as workaholics.

**Stig:** Well, I apply that label from my own experience. That’s how I got started. Maybe workaholism is a bad word. Sometimes I also call it productive compulsion and the phenomenon on a large scale with stone-soup results is distributed productive compulsion.

**Peter:** I see.

**Stig:** I believe that the FSF has tried to establish the GPL as the de facto license for cooperative software. While the GPL is certainly very popular and is perhaps the most popular, I see lots of license fragmentation within the community. There has long been heated debate between those who favor BSD-style licenses and those who favor the GNU GPL, but recently, there have been a number of new source-available packages – notably, TrollTech’s Qt GUI Toolkit – with newer and more restrictive licenses that have served to fragment the community further.

The reason that we’re seeing this happen now is because the GPL makes it difficult to charge money for large but incremental changes to someone else’s code. This licensing fragmentation seems likely to continue until we settle on a license which addresses the economic needs of developers.

[Post-interview note: Recently, Open Source(R) has been coined as an umbrella name for all licenses which meet the Open Source definition. The definition is a spin-off of the Debian Project and has been an important step toward making some sense out of all the different licenses presently in use. Richard Stallman, however, has issued a well-considered statement criticizing the “Open Source(R)” label. Read the definition and also read Stallman’s criticisms. The AFPL does not fit the guidelines in the Open Source definition.]

While the AFPL’s terms are better than those of Troll Tech’s Qt, and you’ve still managed to use the AFPL to generate income by clamping down on the “free rider problem” to some extent. Can you talk about that?

**Peter:** The free rider problem is when someone is allowed to package free software in nonfree or less-free bundles, and that’s precisely the area of the GPL that I thought I needed to do something about in making the Aladdin license, so the AFPL originally prohibited all commercial distribution on removable computer media.

After the AFPL had been out for about a year or two, I added an exemption for collections consisting entirely of freely redistributable software. I did this because I felt that even though people doing free software distributions were making a business of distributing my work, their business was close enough to being participation in the nonmonetary cooperative sphere that I felt it was consistent with what I wanted to do. So the AFPL now permits them to sell such collections without a commercial license.

But that’s where I have drawn the line.
Stig: Well, what would happen if your approach were taken to an extreme? What would happen if all software were distributed according to the terms of the AFPL, which permit gratis redistribution? You would no longer be paid.

Peter: Fine with me. I will go out and do something else. Let me be quite clear about this. At this point, I have made enough money from commercial Ghostscript licensing that I can retire. But let me give you a more serious answer for Ghostscript.

Remember that at the beginning of our discussion we talked about how a large part of what makes the whole consideration of free software even thinkable is the fact that the cost of replicating software is essentially zero. The place where Ghostscript is commercially licensed, and is currently making most of my money, is incorporated into things with a cost of replication that is inherently fairly large. Mainly printers.

I can tell you that the number of companies making hardcopy output devices that have licensed Ghostscript is more than ten and less than a hundred. I don’t even know the whole list any more because my licensing business handles that, but I can tell you that at least two quite well-known companies license Ghostscript, or other freely redistributable technology from Aladdin, for incorporation into hardcopy output devices.

Stig: You’re in a unique position because your software has such a compelling tie to the hardware world. That’s how you make your money.

Peter: I’d say that what makes Ghostscript’s position unique is that Ghostscript’s commercial value is as an OEM component. The hardware tie-in is just the most important instance of this. The printer and controller industries are used to paying license fees for the embedded software, so it’s an easy sell. I guess Russ Nelson found something similar with his work on packet drivers, although his work is freely redistributable from the start. That is, as far as I know, he only charges for development, not per unit license fees.

Stallman has proposed that software development be funded by a tax on hardware manufacturers. What I don’t see is the same, but I could make the argument that I am essentially supporting the development of a largely freely redistributable piece of software with a surcharge on larger products of which the software is a component. Most of these products are physical entities because that’s where the money is.

Stig: Unfortunately, that doesn’t work very well in the general case for other authors of freely redistributable software.

Peter: That’s right. It doesn’t. And, in fact, I believe that if you look around at what successful freely redistributable software is actually out there, as far as I can tell, there are no widely used end-user applications that were developed to be freely redistributable.

Stig: There is Gimp.

Peter: What is Gimp?

Stig: Gimp is a Photoshop-like image-editing application. Gimp has an incredibly dedicated following and its growth has been truly a delight to watch. I consider it to be a flagship project in the cooperative software world, at least equal in importance to Apache.

Two students at Berkeley wrote the vast majority of the Gimp’s core. As is the case for Photoshop, much of Gimp’s functionality resides in plugin modules. That makes it relatively easy for lots of different hackers to develop smaller bits of the Gimp without having to learn the whole body of Gimp’s source code. Gimp is also extensible in scheme, and it’s been called the XEmacs of image editors.

Peter: Well, I have to say that puts a dent in one of my theories about the evolution of free software.

Stig: It’s frustrating, though, to see the primary Gimp authors, Peter Mattis and Spencer Kimball, leave the cooperative software community to take “real jobs” because we don’t yet have an economic model capable of enticing them to stay.

I think that we really need a license and/or mechanism that addresses the general case, that encourages both openness and cooperation while also striving to reward all contributors. Doing cool stuff should be rewarded in a tangible way. It ought to be the difference between eating sushi or eating Taco Bell, the ability to take vacations, buy a home, retire.

Really, I think it boils down to optional spending. How do we get people to value the work done by those who make their lives better. How can we do that in an environment where they don’t have to and they don’t necessarily get bombarded with price tags and invoices?

Peter: There is evidence right now that people are willing to pay for software. People do pay for software. And a good deal of it isn’t very good.

It seems evident that if we understand how to structure the world, then there is quite a bit of money around to pay for software creation. What people actually are paying for commercial software now might be considered an upper bound on the amount that society is willing to pay to make software happen.

Stig: Right. People do pay $600 for Photoshop.

Peter: I can think of several reasons why someone would choose a commercial package over a free package: one is presentation content – the menus, the icons, all of the nice little touches, the polish – and making a piece of software that has all of those qualities is a lot more work than making a piece of software that “just sort of does the job.”
Even if the part of the job that the software does, it does well, the polished package is a lot more work. Testing a piece of software thoroughly is a lot of work. Documenting it well is a lot of work. Supporting it is a lot of work.

And I think that in order for people to trust a piece of software to do something that they need (or very much want) it to do, then the perception that cooperative software or free software doesn’t have those qualities needs to be changed.

As a corollary to this, the model of free software which says that you just pay people for the time that they spend on development is just not sufficient. A model is needed that allows people to be paid to do all of the things that I just enumerated: develop the basic functionality, develop the broad scope of functionality, develop the presentation content, develop the documentation, do the support.

I think actually that has a lot to do with why I didn’t think that any polished end-user redistributable applications existed, and now you tell me that there is Gimp.

**Stig:** Well, Gimp’s not entirely polished, but it’s also not entirely done. It is getting lots of testing from the early users, and it has some very good user-contributed tutorials and documentation on the Web.

**Peter:** So when something gets very close, then at that point, I think it becomes much easier to recruit people to help. And also I think, maybe paradoxically, at the beginning it’s easy to recruit people to help to build up the functional mass. It’s that big piece of work in the middle between something that “sort of works” and something that almost works really well.

**Stig:** This gap in developer interest is very similar to an end-user marketing problem for high-tech products. The phenomenon in the marketing world is explored in great detail by Geoffrey Moore in his book, *Crossing the Chasm*.

In the life cycle of a technology product, there is an initial burst of excitement as the technophiles and early adopters flock to the product. That stage is followed by a problematic gap in the growth of the product’s user base, where nobody buys it because it’s still not good enough for the general public and there are no more early adopters to serve as crash-test dummies and pay for the privilege.

That gap is called the chasm.

That’s the point where start-up companies often need unanticipated additional rounds of investment to stay afloat. If more money is invested, then employee stock options become worthless, and the founders generally lose control of their companies. This is where vulture capitalists earn their bad reputations.

In the world of cooperatively developed software, I guess the chasm is that big gap between “It does what I want it to do” for the initial developer(s) and “It does what we want it to do” for everyone else.

**Peter:** I didn’t have that conceptualization before. It’s interesting that someone has identified it and given it a name. But that chasm is why I didn’t think that good, freely redistributable end-user applications would be developed.

**Stig:** What do you think would happen if cooperatively developed software were the norm in the software industry, instead of the exception?

**Peter:** It seems obvious to me that if a lot more software were developed cooperatively, then there would be a lot fewer people supported by the software industry.

I consider this to be a matter of arithmetic. Right now – as you and I have more or less agreed – the total amount of money that is being paid for software is much too high.

**Stig:** I don’t think so. I think that the money spent on software is an indication of the value that society places on software. It’s neither too high nor too low: it just is.

But I do think that a lot of people aren’t getting the level of quality or the level of service that they ought to get. And I think that there is too much unnecessary replication of effort in the software industry. By having less wasted effort, there wouldn’t be fewer people supported by the software industry, but there would be better software.

**Peter:** But this is not incompatible with the notion that far fewer people can be supported by producing what we now think of as software. While the same amount may be spent by society on computer-related services, they’re likely to be different kinds of services.

Right now, as we’ve agreed, the current methods of producing software are inefficient. There are costs for adding things that people don’t want. There are costs accompanying the sale. There are costs for overlapping development. I would argue that there are unnecessary marketing costs. But that money is all going somewhere.

**Stig:** Much of the competition in the software world is not to the benefit of the end-user. Only Microsoft and Netscape like the browser war, for instance. People hate it. And Netscape is losing right now.

**Peter:** Yes, I agree. However, I believe that if Netscape had not chosen to compete with Microsoft . . .

**Stig:** They would be toast.
Peter: They would be toast sooner. Desktop software is really anomalous economically, because Microsoft has such incredible power in that arena that they can do things that throw traditional economic reasoning out the window.

Stig: What do you think would happen if Netscape took the source code to their browser and released it to the public?

[Post-interview note: Since this conversation took place, Netscape has released the source to their browser. See <http://www.mozilla.org/> for details. Netscape’s Mozilla Public License strives to incorporate the best aspects of BSD-style licenses and the GPL. It is less restrictive than GPL and meets the terms of the Open Source definition.]

Peter: It would win. If they released on relatively free redistribution terms, I think that the resulting browser would win.

I was asking myself this morning, Why is it that I don’t hear a lot about cooperative or freely source redistributable browsers?

Stig: It’s because they’ve been playing catch-up. Mnemonic <http://mnemonic.browser.org/> is being built from scratch by students, mostly students in Europe. And Yggdrasil is funding development of some of Arena <http://www.yggdrasil.com/Products/Arena/> now, but they are paying somebody in India to do the work because they won’t (or can’t) pay Silicon Valley wages for the work. Mnemonic is in its infancy, and Arena is hardly mature.

Peter: To get back to the probable outcome of a source release of Netscape’s browser. I get a feeling that there are qualitatively new and different net exploration ideas that aren’t being worked on now because if they were to be worked on, they would have to be worked on within the bowels of either Netscape or Microsoft.

What I would hope is that if the sources for an existing commercial-quality browser were released, that there would be some chance of something better coming out of the cooperative software world than from either of the commercial browsers. That would be my hope, simply because there would be the ideas of so many more people to draw upon. I think that must be part of the reason for Apache’s success.

Stig: Yes and no. I don’t think that Apache is always doing the latest and coolest things. But I also don’t think that all of the latest and coolest features are ones which people really need. In an effort to differentiate themselves from the competition, Microsoft and Netscape add a lot of extra features that perhaps people don’t really care about. New features in Apache are added because people really need them, while new features in the commercial browsers are often driven by marketing.

Sometimes this is good for the public, and sometimes this is bad. It’s bad when the resulting new “features” serve to twist the arms of the user-base. Use Microsoft’s Word 6.0, for example, and it will quietly start to change the file format of all your documents so that you can no longer use older versions of Word to edit those files. They’re basically “corrupted.”

I think that Apache is successful because it avoids these kinds of problems and because its features are user-driven and not marketing-driven.

Peter: Right.

Stig: Let’s wrap up. I have some final questions. First, where do you think that free and/or cooperative software stands now?

Peter: All right. I can give you a very short answer to where I think that free software stands now.

I think free software’s star is rising. I think that a few very visible success stories – like Apache, maybe like Gimp, to some extent like GCC – are establishing free software as a credible alternative to commercial software. I would hope that, as that credibility rises, companies will be willing to start paying to fund the development of that kind of software. More willing than they are now.

Stig: What is the one meme that you would most like to propagate within the cooperative software community?

Peter: I would like a lot more people in the community to be aware of the cost, the difficulty, and the perceived value of having polished software: software that is well tested, well documented, and well supported. To understand that making software that crosses the chasm does take a lot more effort, and that it’s by having the software that has crossed the chasm that you’re going to attract a lot of people to an alternative way to getting nice software produced. Because I think that is really where UNIX lost out. I think that is why Microsoft now pretty much owns the desktop. UNIX hackers, who have the best system technology, never really made a mental connection with the world of the user. If cooperative software development and free software distribution are going to continue to expand and to start creating a change in mind-set, then there has to be a lot more awareness of that chasm and what it takes to cross it. That’s the one point I’d really like to get across.

Stig: Great! It’s been a pleasure talking to you.

Peter: Well it has. It’s been way too much of a pleasure.
source code UNIX on the PC

Application Software: Ports and Packages

By Bob Gray

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

I'd like to thank Mike Durian for helping me keep this article well balanced. Tom Poindexter and Jordan Hubbard also helped with reviewing.

Three times in the period of a week I had to download the latest source files for publicly available applications (gnuplot, glimpse, psutils) and compile them for my six-year-old SGI Indigo. I needed current versions because of new features or bug fixes. The time spent for this maintenance was two hours, but it is a never-ending, thankless, nonbillable chore. I have about 175MB of frequently used binaries under /usr/local, including bash, awk, emacs, ical, exmh, Tcl/Tk/Expect, ssh, groff, pgp, and tgif. Facilities to minimize the effort of keeping up to date are desirable. The Source Code UNIX systems offer help in this area.

On my networked Source Code UNIX Pentium processor, I also rely on these freely distributed programs, but there is much less work in keeping them up to date. On the FreeBSD system, it's simply a matter of going to the appropriate subdirectory in /usr/ports and typing "make." If a current CD is mounted, the sources for the target program will be copied; then the binaries will be built. If the CD is out of date or missing, sources for the program will be fetched from the Internet and a binary built.

Information about current software versions is in the hierarchy /usr/ports. The make program looks in /usr/ports to figure out what is the current version of an application. Then make looks for matching source code on the CD-ROM. If current source code is not available on the CD-ROM, make will attempt to download source code from the Internet. Then the binaries are built and installed. Finally, an entry is made in a simple database tracking what software is installed so that it can later be updated or removed (pkg_delete). Of course, /usr/ports must be kept very current for this to work well, but the small /usr/ports tree is easy to keep up to date using cvsup, a recent CD-ROM, or ftp.

This article discusses freely available application software for UNIX platforms, how to get it, and how to install and maintain it. My first article (online at <www.boulderlabs.com>) gives many reasons for running Source Code UNIX. Having access to a couple of thousand publicly available applications is one good reason. And access to the source code for those applications is a second good reason. Whether you are in a big office with a platoon of system administrators or in a solo home office, it takes a lot of work to keep software up to date. Anything that helps is worthy looking at. Folks working for the advancement of Linux, FreeBSD, NetBSD, and OpenBSD collectively make installing these applications easier for you. They monitor the development and release cycles of the applications, and they work to make installation as automatic as possible. Source Code UNIX distribution CD-ROMS have many of the applications precompiled and ready to go.

I'll talk about the mechanism used to distribute and install these applications for FreeBSD, NetBSD, and OpenBSD. For Linux, the discussion depends on the particular distributor of the CD-ROM set. One of the most popular is Red Hat Linux with its rpm files. FreeBSD uses the term "port" to refer to the source code for an application. The tree /usr/ports starts out as a small hierarchy containing for each of the more than 1,600 application, a makefile, a checksum, a version number, a description and possibly some patch files. The application source code initially does not reside under this tree.
You can think of the Ports Collection as an infrastructure for building any of the 1,600 applications. In FreeBSD, a package is a tar-ball that contains binaries and affiliated files to run an application. A package is similar to a Linux .rpm file or an SGI inst file. (Packages are created by typing “make package” in an application directory of the /usr/ports directory.) Packages are just conveniences so that the end-user doesn’t have to spend time compiling. The FreeBSD CD-ROM set has both ports and packages on it.

This article addresses freely redistributed software. Here we do not discuss the growing body of third-party, commercial applications for our platform. Big-name vendors porting to Source Code UNIX include Oracle, Informix (databases), Corel (WordPerfect, spreadsheet, etc.), Electronic CAD/CAM, and others. There are alternatives to running Microsoft Office! Check out Applixware <www.applix.com> and StarOffice <www.caldera.com/products/staroffice> — both are complete office suites available for Linux. See <www.linux.org/news/index.html> for articles on the growing body of commercial software for the UNIX platform. Linux has the largest number of installed systems (maybe 7 million); the vendors usually port to this Source Code UNIX first. But the 1.5 million FreeBSD installations can immediately take advantage of these binary releases using Binary Linux Emulation. See <www.linux.org/apps/index.html> for a list of commercial software and see <www.freebsd.org/handbook/handbook.html> for more information on Linux Emulation.

The Ports Collection
You might wonder why there is a Ports Collection instead of just putting binaries into the distribution. There is just far too much publicly available software. Most people don’t have room on their disks for all of it. Much of it is specialized and not of general interest. Then there are controversial (religious) issues about editors, shells, Web browsers, etc. By making Ports optional, the core distribution is kept modestly small and people can customize (add to) their systems as desired. Under FreeBSD, there are two basic mechanisms for loading optional application software: Ports and Packages.

Packages are binary distributions, ready to install. They were created from source code fetched by the ports mechanism. To easily deal with these packages we have

- pkg_add – a utility for installing software package distributions
- pkg_delete – to remove software from your system
- pkg_info – what’s on the system

For Linux, the equivalent system is called RPM or the Red Hat Package Manager. See <refus.w3.org/linux/RPM> for details.

The Ports Collection is an elegant example of distributed software cooperation. The authors of the software make their releases available on FTP sites. A few “ports maintainers” monitor the release cycles of this software and modify the /usr/ports hierarchy to reflect current versions. These guys make sure that the virgin software releases will compile cleanly on the FreeBSD system. If necessary, they create some patches in /usr/ports that will be applied before compilation is attempted. This way thousands of end-users don’t have to customize the same software over and over to run in their environments. To build and install a port, change to the appropriate subdirectory of /usr/ports and type:

- make – compile the application
- make install – install it (usually under /usr/local)

NetBSD and OpenBSD use very similar mechanisms leveraged off of the FreeBSD Ports Collection. In general, once some application software builds on one of the BSDs, it builds on all of them.
The Downside of Ports

You should watch the build as it happens. Often the build will display for you to read a file that has important information on something you need to configure by hand before running the application.

I would be remiss if I didn’t talk about some of the negatives associated with the ports. My colleague, Mike Durian, who has a love/hate relationship with port, is keeping me on an even keel in this article.

The application software comes from all corners of the world. Everyone’s development environment is different. Whereas SUN, SGI, Apple, and others have strong control over the hardware and software that gets shipped, Source Code UNIX systems have cobbled together hardware and mix and match software from Timbuktu. This set of circumstances makes it impossible for everything to work automatically all of the time.

With the Ports Collection, you don’t get to choose where things get installed. There are a couple of environment parameters that control some general layout formats, but you really can’t fine-tune things. (We can see a need for some kind of a software registry database.) Sometimes one port depends on another. SUN, SGI, and others have tools that build dependency graphs to ensure that things remain consistent. When conflicts arise, you are given choices on how to resolve them. The ports mechanism is not as sophisticated and consequently can get into more trouble.

There are assumed locations (generally /usr/local) for certain files and problems develop when various pieces of software make different assumptions. A good example of this is the Tcl/Tk package. If you get and install Tcl/Tk by hand then the chances are you will not be able to use any other packages from the ports collection that need Tcl/Tk. There is a rule in the bsd. *port*.mk makefile template that checks for the default installation location for Tcl/Tk and complains if it is found installed there. The Ports Collection wants it installed according to different rules as defined in the ports makefile. You can debate whose installation location is best, but that’s not the point. The point is the ports package installed it differently and incompatibly from the standard package.

Because you’ve tried to make things easier for yourself by using the Ports Collection, you’ve pretty much locked yourself into always using it. It is difficult to mix and match using the Ports Collection with installing and porting things by hand.

Another downside of the Ports Collection is that tons of assumptions are being made for you about the behavior of your software. For example, the MH (message handling) software is difficult and time consuming to install by hand. The Ports Collection makes it much easier. However, during the exercise of hand porting, you learn about the configuration issues. When you use the ports version, you’ve got to hope the maintainer built it with the features you want and need.

You should watch the build as it happens. Often the build will display for you to read a file that has important information on something you need to configure by hand before running the application. It is all too easy to miss these messages in the stream of compile commands.

Mike also points out the less people are forced to port software themselves, the less they learn about other good styles and practices. He feels fortunate that he had to struggle with other people’s software before ports were available. He feels that he is now a much more rounded, experienced programmer.

Bob’s “Can’t Do Without” Software List

I’ve come to depend on a lot of publicly available software. I carry this repertoire wherever I go. All of these are conveniently part of the CD-ROM distribution and the Ports Collection.
My shell is bash. Its filename completions and command line editing motivated me to switch from csh years ago.

Because of patent issues, the once public compress has fallen out of favor and gzip has taken its place. A nice bonus is that gzip attains significantly better compression. GNU’s tar smoothly works with gzip to give you archiving and compression with one command (i.e., `tar -z`).

Written communication is essential for most of us. These tools help me be more effective: `look`, `dict`, and `ispell` help me with spelling daily. Even though it’s not my primary editor, `emacs` is one of my relied upon tools. I save lots of time with its keyboard macro facility. You “record” a set of steps to do something to one line (for example) and then “replay” the macro thousands of times to get the whole file. `groff` gives me type-setting software; Postscript tools allow me to manipulate it: ghostview, `gs` (Note, Ghostscript [gs] also reads PDF files), `nenscript`, `psselect`, `ps2a` lets me get ASCII back.

To handle the mail, `exmh` makes dealing with volumes of messages as painless as possible. The winning feature for me is that messages are “presorted” into various in-boxes. Metamail (MIME), glimpse (searching), and `popp` (encryption) are built in.

The need for security increases as our computers become more networked. PGP is a great way to communicate securely with others using public-key encryption. When you need to securely connect to a host across the Internet, `ssh` is the ticket. You can log into your home machine over an insecure network using one-time passwords à la `skex`. The `sudo` command gives you one time superuser privileges and logs your activity.

My home directory is 400Mb in size now. It’s easy to find things with the indexing tool `glimpse`. Its companion `agrep` is a fast search tool for “approximate” searches when precise regular expression searches won’t find it. `wget` is a batch retrieval program for fetching files from the World Wide Web using HTTP and FTP. You don’t have to sit around clicking buttons to retrieve a hierarchy over a slow link.

When I need to plot data, I turn to `gnuplot`. I frequently use `xv` to view/convert most kinds of images. `pnmplus` (Portable BitMaps) is the “universal” image conversion software.

I handle my spreadsheet needs with `sc`. `expect` automates dealing with interactive programs far beyond what is possible with shell scripts. `gawk` has the features of the updated Awk programs. I find it a great tool for manipulating tables. I schedule my time with `ical`. It has alarm features and repeat dates. `tgf` is my drawing tool. Now that it is hypertext linked, you can make nice interactive presentations and tutorials. Most of us need tools like `Perl`, `rcs`, and `cvs` every day.

**A Tour of the Ports Collection**

On the next two pages I highlight pieces of the Ports Collection – attempting to give a feeling of what is available in the broadest terms. You will need to go to the source to check on the details of these applications. You can use the `make` `search=` facility to help find applications. (See the handbook). Also glance at the README documents in the sub-directories of `/usr/ports`:

- [www.freebsd.org/handbook/ports.html](http://www.freebsd.org/handbook/ports.html)
- [www.freebsd.org/ports/](http://www.freebsd.org/ports/)
A Tour of the Ports Collection

**Astronomy**
satellite tracking and orbit propagation program
star: star field demo
sunclock: shows Earth’s surface illuminated by the sun
xearth: sets the root window to the image of Earth
xephem: interactive astronomical ephemeris program
xhpaim: sets the root window to the moon in its current phase
xtide: harmonic tide clock and tide predictor

**Audio and Multimedia**
sox: Sound Exchange – universal sound sample translator
rsynth: speech synthesizer
cddbank: Internet CD database server
xanym: play multimedia material
 mkiso9fs, cdrecord, cd-record: CD players and writers
You will find several audio mixers/editors/players, numerous MPEG tools for encoding/decoding and even MIDI tools.

**Benchmarks**
bonnie: performance test of filesystem I/O
bytebench: BYTE magazine benchmark suite
iozone: performance Test of Sequential File I/O
1mbench: system performance measurement tool
netperf: network performance benchmarking package
tcpsblast: measures the throughput of a tcp connection

**Biology**
babel: conversion program for various molecular file formats
rasmol: fast molecular visualization program
seeview: multiple DNA sequence alignment editor

**C&l Tools**
There are a couple of general purpose circuit simulators, VLSI layout tools, and circuit board layout programs.

**Converters**
btca: encode/decode binary to printable ASCII
undview: program for uu/xx/Base64/BinHex de-/encoding
uulib: library for uu/xx/Base64/BinHex de-/encoding
undview: An X11 interface to uulib

**Databases**
bdb: Berkeley DB package, revision 2
gdbm: GNU database manager
gnats: Cygnus GNATS bug tracking system
mysql: a multithreaded SQL database

**Development of Software**
autotools: configure source code on many UNIX platforms
gdb: developer’s version of GNU debugger
mprof: memory profiler and leak detector

**Editors**
axe: simple-to-use text editor for X
staroffice: word processor/spreadsheet/drawing/Web suite
beav: binary editor and viewer

**Emulators**
hfs: reads Macintosh HFS floppy disks, hard drives, and CDs
macutils: utilities for Apple Macintosh files
mtools: collection of tools for manipulating MSDOS files
wine: MS-Windows 3.1/95/NT emulator for UNIX
x48: HP48sx emulator
xcopilot: Emulator for US Robotics Pilot PDA

**Games**
acm: flight simulator for X11
doom: Id Software’s Doom for Linux
gnuChess: “classic” GNU chess
netris: network head-to-head version of T’tris
quake: QuakeC compiler, for building custom games of Quake
xLife: John Horton Conway’s Game of Life
xmille: X window millie bourne game
xrubik: X-based rubik’s cube
xscrabble: X version of the popular board game

**Graphics**
ImageMagick: display and manipulation of images
gimp: general image manipulation program
jpeg: JBIG’s jpeg compression utilities
mencoder: MPEG-2 encoder and decoder
sane: access to scanners, digital camera, frame grabbers, etc.
tg1: Xlib-based two-dimensional drawing facility
xf86: drawing program for X11
xpaint: simple paint program
xpm: X Pixmap library
xv: X11 program that displays images of various formats

**Foreign Language Packages**
German, Chinese, Vietnamese, Japanese, Korean, Russian
dict: simple english/german dictionary
manpages: German GNU and Linux manual pages
Win: Japanese/Chinese/Korean input method
gb2ps: converts Chinese GB (simple) encoded text to PostScript
Canna: Kana-Kanji conversion system
xshodo: paint tool for Shodo – Japanese writing character
rispell: Russian (KO18-R) dictionary for ISPELL

**Languages**
Java: the latest rage in programming languages
expect: sophisticated scripter based on Tcl/Tk
Tcl: Tool command language
Mail
eem: ELM Mail User Agent
eom: X11/Tk based mail reader front end to MH
pop2, pop3, apop, imap
majordomo: The Majordomo mailing list manager
pine: Program for Internet email and News

Math
calc: arbitrary precision calculator
gnuplot: interactive function plotting program
linpack: Linear Algebra package
ss: curses-based spreadsheet program

Misc
amanda: Advanced Maryland Automatic Network Disk Archiver
chord: produces PS sheet music from text input
dejaguar: automated program/system tester
ical: calendar application
tkinfo: Tk script to read GNU “info” files and display them
tkman: Tcl/Tk based manual browser
xinvest: personal finance tracking and performance tool

Net
arpwatch: Monitor arp and rarp requests
cap: Columbia AppleTalk Package
cvsup: network file distribution/update system for CVS depot
dnswalk: DNS debugger – analyze a zone transfer
ftptool: graphic FTP shell based on xview
netatalk: file and print server for AppleTalk network
rsync: network file distribution/synchronization utility
sambe: free SMB and CIFS client/server – talks to Microsoft
scotty: network management extensions to Tcl
traceroute: finds the path of a packet through the network

News
dnews: commercial NNTP server
inn: InterNetNews – the Internet meets Netnews
knews: threaded NNTP newsreader for X
newsfetch: downloads news articles from NNTP server
nn: NN newsreader
ntnp: NNTP with NOV support
slurp: A passive NNTP client that retrieves USENET articles
tin: TIN newsreader (termcap based)
trm: Threaded Read News newsreader

Print
a2ps: formats an ASCII file for printing on a Postscript printer
acrread: views, distributes, and prints PDF documents
apso: lpd magic print filter with autofile type recognition
enscript: ASCII-to-Postscript filter
ghostscript5: Aladdin Postscript interpreter
ghostview: X11 front-end for ghostscript
latex: LaTeX2e – a TeX macro package
latex: document creation system like LaTeX, but smaller
psutils: utilities for manipulating Postscript documents

Security
cfs: cryptographic file system implemented as an NFS server
cpus: system resource cruncher
crack: “Sensible” UNIX Password Cracker
cwtk: toolkit used for building firewalls based on proxy services
rsaref: encryption/authentication library, RSA/MD5/DES
tcp_wrapper: TCP/IP daemon wrapper package
tripwire: file system security and verification program

Shells
bash: GNU Bourne Again Shell
tcsh: extended C-shell with many useful features
zsh: Z shell

Sysutils
cdrecord: creates CDs on a CD recorder
1sof: lists information about open files
mkisofs: creates ISO9660 with (optional) Rockridge extensions
xosview: graphical performance meter

Textproc
catdoc: converts MS Word documents to plain ASCII or TeX
pilot_makedoc: text into the Doc format used by PalmPilots

WWW
apache: popular Apache http server – very fast, very clean
netscape: Web browser
Numerous HTML editors

X11
XFree86: X11R6.3/XFree86 distribution
afterstep: window manager, _NeXTSTEP_ clone
ctwm: extension to _twm_ with multiple virtual screens
explorer: file manager – look and feel of the Windows 95
fw: window manager for _X_
fw95: Win95 lookalike version of the fw2 window manager
viewfax: displays files containing g3 and/or g4 coded fax pages
xpostit: PostIt messages on your X11 screen
xcb: tool for managing X11 cut buffers
xzoom: magnifies, rotates, mirrors the image on the X screen
the security model of JDK 1.2

After the polemics about the JDK 1.1 security policy, the primary JavaSoft objective is to facilitate the work of the applet programmer as well as making Java applications compliant. The aim of the designers of JavaSoft is to provide multilevel security mechanisms as outlined in following the steps.

Fine-Grained Access Control: This widens the concept of granularity, which, to be well implemented, requires considerable effort at the programming stage to optimize the development personnel, with experience not only in programming, but also in information security. During a recent unofficial conference held in Rome, Italy, representatives from the computer security industry, academia, and Sun Microsystems jokingly observed, “How much is the developer really in possession of all these qualities to prevent costing 200 million lira per year?” The likelihood, at least in Italy, is very low! This is the same argument that the designers of Sun Microsystems must have come up with when drawing up the new security analysis model.

Easily Configurable Security Policy: Though this was included in the previous release of the JDK, it was not easy to use. The main improvement to this problem will be to allow the developer to implement personalized and flexible security policies without programming.

Easy, Extensible Access Control Structure: Many of the criticisms brought to the designers of the sandbox have commented on the lack of configurability of the Security Manager for creating new access permissions. To address this criticism, it was necessary to program a new check method and add it to the Security Manager. Few developers were available to solve this problem, which represented a potential leak in the management of the strategic security system.

The Evolution of the Security Model
Related to the general analysis model, there are different complementary approaches to Java security, especially concerning the applet layout and application. One of the recently proposed techniques is “Java Stack Inspection.” The principle here refers to undocumented security procedures that can be executed automatically, allowing a more flexible standard policy for the sandbox. Automation of the procedures is possible by rewriting all the Java classes to execute an additional step that will invoke the check procedure. This is called security passing style.

The Stack Inspection has been adopted from the browser manufacturers (Netscape and Microsoft) into JDK 1.2. The general principle of Stack Inspection has been implemented in Netscape Navigator version 3.0 onwards. Without going into great detail, Java Stack Inspection is based on the principle of “Privilege assignment,” based, in turn, on the following:

- enablePrivilege()
- disablePrivilege()
- checkPrivilege()
- revertPrivilege()

These elements are set up to guard the “dangerous resource,” a term used to describe resources that must be protected. This is reminiscent of a recent technique criticized by some people for being too complex.

Another study has been undertaken in order to solve the problem of multiprocessing. This is especially the case for Java-based solutions in the networking field, where the demand for the Java platform is in the multiprocessing field for multiusers. There have
been some attempts to solve this problem in JDK 1.2 beta version in order to provide security in the execution of the mobile Java code and increase access to more users.

As previously mentioned, the execution of “risk” actions from the part of the .class file is substituted in the first place to the security manager of the JVM, which analyzes the calls according to the cases and allows or disallows the execution. JDK 1.2 has an alternative process evolution of applet checking. To create a distinction between local code and remote code, policies are set before deciding the work. This is also the case to determine whether or not the code is from a trusted source based on a digital signature. In the multiprocessing field, this operational principle will be taken as a base to combine the source-based policies with the user-based policies.

Conclusions
On one hand, many developers are disposed to think well of the developments in the new model of Java security. Others see the developments as an acceptance of defeat to those who have preached about the insecurity of Java.

On the other hand, it appears that the new “policy and permission-based” security model is an intelligent simplification of the programmer’s work, but for the developers, the “signature-based” method has been poorly considered. For some, who remember the failure of Microsoft’s Authenticode to secure ActiveX, JavaSoft is an unloading of responsibilities for developers. However, sole responsibility for applet security now lies with the developers.
java performance

Wrapper Classes
In Java, all class types are directly or indirectly derived from the superclass Object. Primitive types such as int or double are not class types and do not have this property, but they do have class type wrappers that can be used to represent the primitive type and hence plug in to the Object hierarchy. For example, the wrapper for int is Integer:

```java
Integer x = new Integer(37);
Object p = x;
```

In this example, I've taken an integer constant (37) and created a class object instance to represent this value. The object instance can be assigned to an Object reference and manipulated in various ways, such as representing a set of values in a Vector. Wrappers also offer other services beyond representation, such as conversion between strings and values of a primitive type (for example, `Integer.parseInt()`), and representation of the minimum and maximum values of a primitive type (such as `Integer.MIN_VALUE` and `Integer.MAX_VALUE`).

Wrapper Overhead
Wrappers have some overhead associated with them. They offer generality at the cost of inconvenience in digging out the value:

```java
int i = ((Integer)p).intValue();
```

There are some space costs as well. To more closely examine this latter point, we can write a program that allocates a Vector of int or int wrappers, and compute the size of each Vector element:

```java
public class test1 {
    public static final int N = 25000;
    public static long freemem() {
        return Runtime.getRuntime().freeMemory();
    }
    // ints without wrappers
    public static void test01() {
        long start_mem = freemem();
        int vec[] = new int[N];
        for (int i = 0; i < N; i++)
            vec[i] = i;
        long end_mem = freemem();
        long n = (start_mem - end_mem) / N;
        System.out.println("bytes per element = " + n);
    }
    // ints with wrappers
    public static void test02() {
        long start_mem = freemem();
        Integer vec[] = new Integer[N];
        for (int i = 0; i < N; i++)
            vec[i] = new Integer(i);
        long end_mem = freemem();
        long n = (start_mem - end_mem) / N;
        System.out.println("bytes per element = " + n);
    }
}
```
public static void main(String args[]) {
    test01();
    test02();
}

This program uses a system method freeMemory() that returns the amount of memory currently free. Note that this technique for determining memory use per element should be employed very cautiously, given the vagaries of garbage collection and so on.

When we run the program, it reports 4 bytes used per element without wrappers and roughly 16 per element with a wrapper. A wrapped double reports as 24 bytes per element, with the actual double value as 64 bits (8 bytes). The space overhead of wrappers goes to support system features such as garbage collection. Allocating a dynamic object in Java has similar overhead to allocating a block of space in C using malloc().

Wrappers have considerable advantages in that primitive types can be treated in a way similar to class types. See, for example, the discussion on page 243 of Arnold and Gosling's book The Java Programming Language (Addison-Wesley). But they do take extra space and time, which may be an issue in some applications.

**Arrays of Primitive Types**

If you're interested in manipulating arrays of primitive types directly, without use of wrappers, newer versions of Java (1.2) offer basic sorting and searching capabilities for such arrays. For example, to sort an array of integers, you can say this:

```java
import java.util.Arrays;

public class test2 {
    public static void main(String args[]) {
        int vec[] = {1, 3, -18, 59, -67, 23, 97};
        Arrays.sort(vec);
        for (int i = 0; i < vec.length; i++)
            System.out.println(vec[i]);
    }
}
```

Arrays.sort() takes a reference to an array. The array includes an internal length descriptor, and is sorted in ascending order upon return. java.util.Arrays is a class loosely associated with the set of Collection classes offered in Java 1.2. Arrays.sort() can be applied both to arrays of primitive types, and to arrays of standard wrapper objects such as Integer and Double. The sorting logic in the Arrays class is duplicated for each primitive type.

Note that a language like C++ solves the wrapper/sorting problem in a different way, by use of a sort() template and by assuming the existence of a built-in or user-defined operator like “<” that is used to order array elements, both for elements of primitive type and for elements that are class objects. Templates and operator overloading represent a more general solution to the problem of sorting arrays of elements, but at the same time are more complex and harder to understand and implement.
using java

Java RMI, CORBA or COM?

Java has been adding new packages to each release, and the JDK1.2beta3 release is no exception.

Most of the time the introduction of new packages is driven by the need to fill a gap. Some examples of packages that were not in the core release of the 1.0 release but were introduced later are JDBC, security, beans, and Remote Method Invocation (RMI).

The inclusion of CORBA (Common Object Request Broker Architecture) services in this release raises some pertinent questions such as "Should I use RMI?" "When should I use CORBA with Java?" "Should I use DCOM (Distributed Component Object Model) instead of RMI or CORBA?"

The answer to these and other related questions is not simple; however, we can ease that burden by knowing more about each of these technologies so we can make intelligent decisions based on what we know. The JDK1.1 release supported RMI, so there was no compelling reason to package CORBA and give it away for free. Or was there?

It is likely that Sun and OMG (Object Management Group, consisting of over 800 companies) had many discussions before this decision was made, and ultimately it does not matter why this decision was made. What does matter is which technology the user endorses.

This article provides a taxonomy of the "good," the "bad," and the "ugly" of CORBA, RMI, and DCOM. Although DCOM is a Microsoft product and currently is unavailable for UNIX platforms, some vendors have announced plans to support DCOM under Solaris, HP-UX, and Irix (and I would not be surprised if Linux support shows up shortly thereafter).

Synopsis

Traditional approaches to executing code residing on machines across a network have been confusing as well as tedious and error-prone to implement.

A useful way to look at this problem is that some objects reside on a machine across a network, and you can invoke methods on that object by sending messages to that object and have it return results as though it were running on your local machine.

The main goal of all three technologies is to provide an infrastructure to enable writing distributed applications with relative ease, although they do so in different ways.

In summary, Java RMI comes from JavaSoft, CORBA is a specification resulting from the OMG, and DCOM comes from Microsoft Corporation.

It is no secret that the CORBA specification is in response to Microsoft's COM and DCOM technology. Given that all of these technologies can be used on both UNIX platforms, we are faced with the dilemma stated previously, namely, "Which technology do we use?"

All of these technologies are similar insofar as they have some kind of registry for registering objects and use an interface definition language to generate stubs for the client code. I am not going to go into the details.

Java RMI

Remote invocation is nothing new. For instance, C programmers have used RPC (Remote Procedure Call) semantics to execute a C function on a remote host. What makes RMI different is that in Java it is necessary to package both data and methods
and ship both across the network (RPC works on data structures primarily), and the recipient must be able to interpret the object after receiving it.

JavaSoft has announced that it will be making efforts to integrate RMI and CORBA but it is not clear what this really means.

So let’s take a quick look at RMI.

The Good:

1. It is very easy to use.
2. Remotable interfaces have a special exception.
3. It supports object-by-value.
4. Versioning is built into serialization.

The Bad:

1. Java call semantics are changed so that thread identity is not maintained.
2. Callbacks are blocked in synchronized methods.
3. It is not always intuitive.
4. It is not available to other languages.

The Ugly:

1. There are limited development tools.
2. Clients need access to the latest stubs.
3. Performance can be slow as you scale.

CORBA

The Common Object Request Broker Architecture is probably the most ambitious and important middleware project ever undertaken in the industry. The OMG consortium (Sun was one of the founding companies) represents a broad spectrum of the computing industry (with the notable exception of Microsoft).

The following is a summary of CORBA.

The Good:

1. It is an architecture for system composition.
2. It has a standard terminology for concepts.
3. Declarative interface separates the interface from the implementation.
4. It provides mappings from IDL to C, C++, ADA, SmallTalk, and Java.
5. Because it was designed for distribution first, there is support for evolvable and marshallable data.
6. It supports design portability.
7. It is scalable for large systems.
8. It supports standard interoperability protocols.

The Bad:

1. There is no inheritance for Exceptions.
2. Inheritance causes problems in versioning, so objects cannot support two versions of the same interface.
3. IDL is not internationalized.
4. There are divergent security mechanisms (kerberos, SSL).
5. There are basic standard service, but few advanced services.

What makes RMI different is that in Java it is necessary to package both data and methods and ship both across the network, and the recipient must be able to interpret the object after receiving it.
Microsoft is serious about distributed Java objects, but its solution is not based on the JavaSoft RMI/CORBA model. Instead, it is pushing for DCOM to be the alternative for its distributed Java.

The Ugly:

1. C++ mapping has complicated memory management rules.
2. There are limited developer tools (usually just an IDL compiler).
3. The limited concurrency model means there is no standard for thread priority, deadlines and timeouts.
4. Scalability can be an issue if design is not well thought out.

DCOM
The Distributed Component Object Model is Microsoft's solution for supporting distributed computing with objects. Another way of looking at it is the DCOM is the foundation for Microsoft's Internet and component strategy. For instance, an ActiveX control is a DCOM object. The VI++ package includes language bindings for DCOM. In fact, Microsoft is serious about distributed Java objects, but its solution is not based on the JavaSoft RMI/CORBA model. Instead, it is pushing for DCOM to be the alternative for its distributed Java.

The following are the salient points of DCOM.

The Good:

1. There are lots of tools, books, and developers.
2. It solves many of the idl versioning problems by separating out the interface from the implementation.
3. There is good integration of automation objects with VisualBasic and Java.
4. There is a good set of compound document interfaces.
5. Microsoft depends on it working.

The Bad:

1. There is minimal support on non-Microsoft platforms. Although some UNIX vendors plan to support DCOM, this technology has not been proven stable enough to gain acceptance in the UNIX world if it were available today.
2. Automation versus MIDL type system (fairly restrictive) is an issue.
3. It cannot passify idle service so a reference to a service must stay in memory.
4. It is hard to keep registry consistent.

The Ugly:

1. Language mappings do not provide for automated UUID changes when an interface is revised.
2. Generated code is intermixed with user code. Headers are neither C or C++.
3. Reference counting is a problem; it is easy to have one too many or one too few releases.
4. The client has to choose the interaction model (CreateInstance, GetObject, Monikers). It is not clear which is the appropriate one for a given case.

What Does This All Mean?
Now that we have all this good stuff out in the open, what does it all mean? In order to do justice to that question, it is necessary to understand the way component-based system architectures using Java are being developed.

One key consideration is to create an evolvable systems architecture. This means that, by definition, the architecture will have to deal with legacy code for obvious reasons of reliability, high development cost, and other important considerations.
One important issue is running the same code on as many platforms as possible. If the requirement is that no recompilation should be required when moving applications to various platforms, then the Java/CORBA solution may be more appropriate than DCOM.

If the application requires interaction with DCOM objects, then, although it may still be reasonable to use Java and CORBA with a bridge technology to interface with DCOM, it may be a better choice to go with a pure DCOM solution.

If you anticipate that your distributed application is going to be written entirely in Java (for rapid prototyping, perhaps), then it is probably a good idea to use RMI for its ease of programming.

It may be necessary to plug in components created by third parties. These components may be based around a DCOM or CORBA interface.

**Summary**
The systems architect is faced with many challenges in defining an evolvable system. The architecture will need to support both short- and long-term goals in the absence of precise information.

The choice of hardware platform can be almost a trivial decision because it may be driven by cost. The software choices are somewhat more perplexing and critical. The hidden cost of maintaining software is a compelling reason to evaluate the various options as objectively as possible.

RMI, CORBA, and DCOM are all examples of middleware. The development of middleware will continue to play an important role in enabling the development of evolvable systems.

The JDK1.2 beta release supports CORBA and RMI. The decision by JavaSoft to go with CORBA was necessary to not compete with the OMG. Given that DCOM is fundamental to Microsoft’s future, it will be pervasive and is part of every Windows OS.

There are no simple choices, and the focus of this article has been to draw attention to the advantages and disadvantages of each technology to make it a little easier to define a system.

The ultimate responsibility falls on the shoulders of the systems architect, who must consider the options and make the best possible choice to meet the requirements of the project.
by Rik Farrow
Rik Farrow provides UNIX and Internet security consulting and training. He is the author of UNIX System Security and System Administrator’s Guide to System V.

musings

We live in a rapidly changing environment. Just yesterday, I was watching little puffs of white cloud soar up into the sky above the Mogollon Rim. As these clouds moved higher, they left a smaller, mushroom-like stem below them. Once aloft, they continued to expand, turning from brilliant white to a threatening gray. Within 90 minutes of their first appearance in a cloudless blue sky, thunder was rumbling. The band of young thunderstorms rolled toward the southwest, now visible only on radar (courtesy of Internet weather sites).

Our work environment is changing rapidly as well. When I began writing my UNIX system administration book in 1985, networking was rare, and certainly not the norm. Today, it is rare to find a computer in an organization that is not connected to a network. And organizations themselves are connected by networks, both public and private.

Banks have relied on private networks for years, using encryption to add to the privacy of the data exchanged. The algorithm used, DES, was deemed both secure and safe. Recent collaborative projects using the Internet had shown that “brute forcing” a 56 bit DES key takes weeks of effort involving as many as 14,000 computers. An article about the first DES challenge appeared in the Security special issue of login: (May 1998). A second challenge, completed in February of this year, was testing possible keys at a peak rate of 34 billion per second.

Brute Force
The term “brute force,” when applied to encryption, means to search the key space, trying each key, until the correct key is found (more on that later). The key space doubles each time another bit is added to the key length. A 16-bit key has a key space of 65536 keys, 17-bits 131072 keys, 18-bits 262144 keys, and so on. There are 72,057,594,037,927,936 potential 56-bit keys, and trying each of them can take a long time.

It is the length of time spent trying possible keys that provides most of the protection of DES encryption. In early 1997, cracking a 40-bit (RC5) key took only 3.5 hours. The first successful DES challenge <http://www.frii.com/~rcv/deschall.htm> took over four months to search 18 quadrillion keys (almost exactly one quarter of the key space). On the average, a brute force approach will discover the correct key after searching 50% of the key space. In other words, it might have taken as little as several seconds, or as much as 16 months, to discover the correct key during the first DES challenge.

Although the point of the challenge was to underscore the weakness of DES, members of the Department of Justice and Louis Freeh, the Director of the FBI, used the challenge to promote the strength of DES (see <http://www.frii.com/~rcv/hir-hear.htm> for a censored version of Freeh’s testimony). The US government has opposed the unencumbered use of encryption and its free export under grounds that such availability would aid the “four horsemen”: terrorists, drug traffickers, child pornographers, and mobsters. Others have argued that restraint of encryption technology amounts to a denial of the right to free speech. So far, the government has yet to prove the strong encryption has significantly impacted any investigation.

Old Sun Hardware
While the government restricts free trade under the guise of potential criminal investigations, the “others” have not been idle. The Electronic Freedom Frontier <www.eff.org> sponsored a project to develop hardware to crack DES in less than a week, which suc-
ceeded in cracking a DES key in only 56 hours (again after a search of approximately 25% of the key space).

Arguing against the government’s position, many crypto experts have pointed out that the way to brute force DES is not by gangning many general-purpose computers together, but by building specialized hardware. The government certainly has the means to design and build DES crackers, and the EFF showed that, for a cost of $220,000 and some donated programming time (about two weeks), they could do it too.

The body of the “device” is an old Sun 4/470 chassis, a cabinet about the size of a two-drawer file cabinet containing 12VME bus slots. I remember these partially for their size, but mainly for their banks of noisy fans. Each VMEbus board contains 64 custom ASICs (Application Specific ICs), and each ASIC contains 24 “search units.” The EFF group used two chassis, for a total of 24 boards, 1,536 search devices, and a PC to handle control and for checking potentially correct keys.

One of the issues in cracking cryptotext is, how do you know when you have found the correct key? Although the EFF device can handle other attacks, the specific challenge involved a known plaintext attack, that is, that a portion of the plaintext is known. Each search device decrypts eight bytes of cryptotext with the candidate key, and sees if the decrypted result matches the known plaintext. To imagine how this might work in general, if the message is known to be ASCII, the known text will contain only alphanumerics and punctuation, a set of about 56 out of 256 different bytes.

You can learn how to build your own DES cracking device (for only $130,000, as you will not have to design your own ASICs) by reading a new O'Reilly & Associates book, named appropriately, Cracking DES, Secrets of Encryption Research, Wiretap Politics, and Chip Design, and written by John Gilmore and Paul Kocher. You can also view this book online (which will save you having to edit OCR scans of the source code) at <http://www.replay.com/cracking_des/toc.html>.

While building your own DES cracker may not be how you plan on spending your next paycheck, the EFF has pointed out just how easy it can be to recover data encrypted with DES. In earlier work by Michael Wiener of Bell Northern Research in 1993, on paper, a design of a DES cracker would be able to brute force DES keys using a $1 million machine in less than four hours.

Today, our governments probably use such machines to monitor fund transfers, especially international ones. One of the implications of using stronger encryption (longer keys) is that it will become much harder for governments to track the flow of money, which will affect money launderers, as well as the very rich (who might be inclined to avoid taxes by using offshore havens). As far as worrying about having your credit card stolen by some local hackers who have sniffed DES-encrypted transactions is concerned, well, I think they will think of better ways to spend the $130,000.

The Answer

Attacks involving captured communications still occur daily. Although the heyday of password sniffing was in 1994 (when all of the big ISPs had problems), password sniffing still goes on today. Although an ISP is the best place to locate a password sniffer (you can see lots of interesting traffic, as well as passwords which you know will work through a firewall), internal networks are vulnerable as well. The 10pht’s-password cracking tool is designed so that it can guess passwords based on the challenge-response pairs sniffable from networks of NT systems <http://www.10pht.com/10phtcrack/download.html>.
Although many vendors boast of being IPSEC compliant, most vendors’ products will not work with other vendors’ products. When products comply with a standard, it should imply interoperability as well.

One answer would be to encrypt all communications between systems. This solution would please CPU vendors, whose chips have become so fast that they need more work to do (encrypting data would fit the bill). More importantly, encryption would really do a lot to improve network security.

Security Architecture for Internet Protocol, better known as IPSEC, will someday make encryption commonplace. IPSEC already exists (the RFCs date back several years), and there are working implementations of it. One problem with IPSEC (besides government roadblocks to better encryption) is the lack of a working PKI (Public Key Infrastructure). Today, getting IPSEC to work relies on two factors—choosing products which interoperate and manually managing keys.

Although many vendors boast of being IPSEC compliant, most vendors’ products will not work with other vendors’ products. When products comply with a standard, it should imply interoperability as well. One of the main areas of incompatibility has to do with key management. If there were a single PKI, as well as a popular implementation, then vendors would be forced to comply or be left stranded with their private standard.

We might be getting some help from an unlikely source. NT 5 includes IPSEC support and also includes Kerberos and support for certificates services. At the same time, MIT, as well as large financial firms, are pressuring Microsoft to make both their Kerberos and IPSEC implementation interfaces public, and therefore (at least hopefully) interoperable. It might still take years, but we may “soon” see desktop-to-desktop and desktop-to-server encryption as NT 5 installations become common (that’s why I said years).

I heard about this while at the LISA-NW conference in Seattle this August. As usual, the first week in August is warm and dry there, but don’t tell anybody. Microsoft announced that NT 5 was now some 35 million lines of source (double the size of NT 4). They also announced innovations, such as drive mirroring and RAID, as if they were newly discovered. Many attendees marveled at this. Someone asked if it was possible to script the mirroring interface (for unattended backups and such); the answer, alas, was a qualified no.

Microsoft also announced a UNIX compatibility toolkit, based on MKS-UNIX tools for Windows. The toolkit includes MKS’s version of the Korn shell, which prompted a gray-haired man, wearing a T-shirt with his own name on it, to stand up and approach a microphone. This person began to explain to the Microsofties that the MKS Korn shell was not compliant with even half the specifications in the two books published so people can write compliant Korn shells. The Microsoft engineer attempted to argue that their Korn shell was compliant, until someone pointed out that the man he was facing was Dave Korn.

UNIX Forever?

Many strange things have happened in my life. One that had powerful effects on me was the success of UNIX, changing my life in many ways. Instead of becoming an MS-DOS or Windows mechanic, I chose to work with UNIX, which has enriched me in many ways. But there were strange side effects, as UNIX became more common.

For several years, I consulted for UNIX World magazine, acting as its technical editor. As UNIX became more mainstream, the magazine found itself in trouble. The traditional “sponsors” of the magazine had stopped buying big ads and were instead placing ads in Business Week and Time. UNIX was no longer a niche market, and the niche magazines suffered. So did the big UNIX shows, with UNIX Expo folding first, and UniForum just last year. Note, however, that UNIX Review survived through evolving.
The rise of NT has been important for UNIX as well. NT has helped push UNIX into mission-critical servers. As a side effect, I have found that my UNIX security courses are more in demand than ever before. Thank you, Microsoft.

The world is a strange and quickly changing place. Today, the scuds that boiled up from the Mogollon Rim have failed to produce thunderstorms, but instead have dotted the sky with fluffy white clouds. It is difficult to see very far into the future, so I will check the radar for the Southwest, and see if there aren't storm clouds hidden behind those distant peaks.
The Future of POSIX
Those of you who follow this column regularly know about the debate that has been continuing in the IEEE Portable Applications Standards Committee (PASC) for the last year or so: What is POSIX? Does it have a future? If so, what is it and how do we get there?

In July this year, the ad-hoc group that was set up to investigate and report on these issues produced its final report. This report is reproduced below. Closer cooperation between the three groups that have up until now been responsible for producing standards with overlapping scope now looks certain to happen.

The three groups, PASC, ISO JTC1/SC22/WG15, and The Open Group (TOG) met together in early September to start work on a revision to POSIX.1 and POSIX.2. The idea is to produce a single document that will define the core of the Single UNIX specification, and replace POSIX.1 and POSIX.2 as they are today with a more up to date and complete standard. This is the first meeting of the study group proposed in the report. This study group is responsible for planning, scoping, and initiating a new joint working group to produce these new standards. This represents the future of the core POSIX work.

Much of the recent POSIX work has been in relation to the realtime extensions, which are still recognized as vitally important, and a place where PASC expertise is still essential. In order to finish those realtime extensions not yet completed, POSIX.1a and POSIX.2b must clear the way for them. Both of these documents are in ballot at present, and look likely to complete within the next year. Apart from providing a basis for the other realtime extensions, these documents will add several classic UNIX features, such as symbolic links, to the existing core.

Let’s hope that we can now stop wondering quite so much about what the future holds for us, and instead get on with implementing it!

The Report of the Ad-Hoc Committee
At the April, 1998 meetings PASC, WG15 and The Open Group (TOG) all committed to seeking answers to the procedural, political, financial, and intellectual property issues which must be addressed to enable the coordinated development and maintenance of POSIX standards. However, proposing solutions, or even identifying who should be involved with specific negotiations, has proven to be very difficult.

The April 1998 preliminary report (N719) from the POSIX Coordination Ad-Hoc Committee identified three “First Principles”:

1. For each specification we must develop a common point of responsibility for development and maintenance of that specification.

2. Methods must be established to ensure technical coordination of logically related groups of specifications.

3. We must establish a sustainable process that ensures that adopted standards are maintained.

Notes:
1. There must be no more than one working group per specification.

2. Working group membership must be open to participation of persons from PASC, WG15, and The Open Group.

3. We must not develop competing specifications.
4. The anticipated lifetimes of standards range from five to thirty years.

This Final Report proposes some basic resolutions intended to define the way forward and provide a “stake in the ground” for resolving the numerous issues which must still be addressed.

In previous debates we have agreed that the existing “rolling amendment” process for POSIX standards is no longer acceptable. We have further agreed that we must establish a step-wise revision process. In January 1998, we took initial steps toward that goal by resolving to limit possible amendments to POSIX.1 and POSIX.2 to some of the existing projects and corrigenda. We further resolved to develop all other POSIX standards as stand-alone specifications that could, as appropriate, be included in a future revision of the base standards.

The 1984 /usr/group Standard was the first effort to define a single, common specification for the various UNIX implementation versions which had evolved in the commercial and academic arenas. This standard, along with AT&T’s System V Interface Definition, were used as the basis for the IEEE/TCO S efforts to develop POSIX standards. In the same time frame, X/Open, in close cooperation with the POSIX committees, developed its series of Portability Guides (XPG). The extraordinarily close cooperation between these two efforts has resulted in the incorporation of all POSIX.1 and POSIX.2 concepts and features (including many which were contributed by TOG) in TOG’s Single UNIX Specification (SUS). Today, the POSIX.1 and POSIX.2 base standards and the SUS are very closely aligned. The SUS represents valuable input to the revision process proposed below.

We urge PASC and the WG15TAG to endorse the following proposals at their July 1998 meetings, the TOG liaison to obtain TOG concurrence, and WG15TAG to submit this proposal to WG15 for its approval. It is essential that these agreements be achieved by the end of July, 1998. The SC22 TAG can then provide the information to the SC22 plenary in August when SC22 is expected to take up the issue of the future of WG15.

There are a number of issues that led to this proposal:

1. There has been a very significant decline in participation in PASC working groups (especially POSIX.1 and POSIX.2) for two primary reasons: (1) we have accomplished most of what we originally set out to do with the base standards (POSIX.1 and POSIX.2); and, as a result, (2) most of the industry participants have shifted their technical expertise to evolving and maintaining the SUS.


4. The “namespace reservation” issue must be resolved to enable other POSIX.1 amendments to move forward.

5. Further delay in resolving these issues, or failure to clearly define a way forward, will result in the near term loss of many of the remaining industry technical expert participants in PASC.

6. We must re-establish a dedicated, critical mass of participation.

In addition, there are numerous unresolved issues that must be addressed if we are to achieve the three “first principles” that we identified at the beginning of this report. Some of these unresolved issues are:

1. All interested participants must have visibility into the standards development and balloting processes. This includes all ballot objections and how they are resolved. One serious impediment in this area may be nondisclosure agreements.

2. There must be a way of balancing the interests of various affected groups. An example method of achieving this is the IEEE requirement for appropriate levels of representation from Producers, Users, General Interest, and Academic participants.

3. There are a number of difficult issues surrounding the ballot process (e.g., who gets to participate, whose votes are counted, the time frame for voting, etc.).

4. With respect to The Open Group processes, there are concerns about the disenfranchisement of individuals and small organizations.

5. A list of specific issues and problems regarding the Procedures and Processes of each organization has not been generated. This should be done and each issue addressed.

6. Specific IPR issues must be identified and submitted for resolution to the appropriate leadership of each organization.

Hammering out the details of these and other issues will continue to be a very major and difficult problem. The ad hoc committee stresses the importance of recognition by all parties that “the devil is in the details” and there are many details remaining to be addressed. The concerns raised by the lack of resolution of these details must be overcome if progress is to be made.

Recognizing the “realities” of the current situation, but also acknowledging the continuing importance of POSIX standards in the quest for truly open systems environments, we therefore re-affirm the Issues and Recommendations presented in the preliminary ad hoc report and further propose the following actions:

1. PASC should adopt a Resolution to immediately create a Study Group with open membership to initiate a joint revision of ISO/IEC 9945-1, ISO/IEC 9945-2, IEEE Std 1003.1, IEEE Std...
1003.2, and the appropriate parts of The Open Group SUS. This revision should result in a common standard(s) that must be completed within the next three to five years. (Note: See IEEE/PASC Resolution 9801-03 for additional applicable information about possible amendments.)


3. Based on the experiences gained from this process, similar collaborative efforts for other POSIX standards should be initiated as appropriate.

4. PASC should adopt a Resolution that PASC will not unilaterally produce standards that conflict with the SUS or the ISO/IEC 9945-1 and 9945-2 standards.

Chair’s footnote:
This report is the result of approximately 12 hours of face-to-face meetings, ongoing electronic discussions over the last six months, and many hours of private discussions on the part of approximately 20-25 participants from all three organizations. The frank and open discussions of the ad hoc committee should be continued in all three organizations. I extend my personal thanks and appreciation to everyone who participated in these discussions.

Roger Martin, chair, POSIX Coordination ad hoc Committee.

Project Management
The PASC working groups have been responsible for delivering many standards projects successfully over the years. Occasionally, however, we embark upon a project that ends up deadlocked for any number of reasons, and fails to make any headway for an extended period of time. Although these projects have their own constituency who are committed to the results of the project, if that project cannot complete in a timely fashion, those constituents will end up looking elsewhere. From time to time, projects end up so helplessly deadlocked that they cannot make sensible progress. Additionally, standards projects, at least in the IEEE, are staffed entirely by volunteers, and these volunteers sometimes lose the time that they dedicate to progressing standards work to other projects their employers demand.

As a result, some standards projects end up in a stagnant form, with no visible progress even to those who are looking closely. It serves the community as a whole if such stagnant projects are actually withdrawn, rather than remain unfinished. This year, we have withdrawn sponsorship for POSIX.1e and POSIX.2c (security), and at the July meeting for POSIX.14 and POSIX.18 (Multi-processing Profile and “POSIX” profile) were also withdrawn.

Each quarter, projects that seem to have remained static for more than a year will be given a warning that unless they can show some sign of life, they will have their sponsorship withdrawn at the next meeting. Of course, withdrawal of sponsorship is not in itself a death sentence. If those concerned are actually doing something worthwhile, they can always reapply for sponsorship.

Common Information Model for UNIX
The Open Group’s System Management working group (TOG SysMan) has spent much of the last year contemplating what to do in the context of TOG’s “IT Diadrome” initiative. It must be said that very little real progress has been made on this; lots of words and pictures, an architecture document, but little that will really effect anybody! One thing that has emerged from this is a commitment to making the Desktop Management Task Force (DMTF) Common Information Model (CIM) a keystone for future system management work. Information about CIM is available at http://www.dmtf.org/cim/CIM_HOME_Page.htm.

CIM is a conceptual information model for describing management that is not bound to a particular implementation. This allows for the interchange of management information between management systems and applications. This can be either “agent to manager” and “manager to manager” communications which provides for Distributed System Management.

This is the first time in this industry that a common method of describing management information has been agreed and followed through with implementation. Other efforts have failed because of the lack of industry support. The model, because it is implementation independent, does not provide sufficient information for product development. It is the specific product areas, applications, system, network, database and devices and their product specific extensions that produced workable solutions.

In July, at Miami, SysMan agreed to sponsor a project to provide UNIX-98 extensions to the CIM common system schema.

CIM schemas are described in Managed Object File (MOF) format. The CIMUNIX project team are writing a MOF description of the things that are in UNIX-98, but are not represented in the CIM common system schema. For example, there is a UNIX_FileSystem class, based on the CIM_FileSystem class, but adding features such as the total number of Inodes and Free Inodes (known as “slots” in UNIX-98, to avoid implementation specific language).

A variety of vendors are producing tools that use CIM already; standardized UNIX extensions will make these tools even more powerful and useful in managing a heterogeneous network. CIM is a part of the DMTF Web Based Enterprise Management (WBEM) initiative; in fact at present, CIM is the only approved WBEM specification.
It’s clear to me that Blake’s “dark Satanic mills” must be at work printing books. The spate continued during the summer (nearly 100 publishers’ announcements of Windows98 books, for example). I, however, having completed a large project (see end of column), read and browsed my way through something I first looked at 30 years ago.

### Fundamental Things

There are few books that are “must reads” within the programming community. The three volumes of Knuth’s *Art of Computer Programming* are certainly among this small band.

Since 1968, when *Fundamental Algorithms* appeared, *Art* has been revised and expanded. Over the past year, Addison-Wesley has brought out third editions of volumes 1 and 2 (*Seminumerical Algorithms*) and the second edition of volume three (*Sorting and Searching*).

The amazing thing, to me, is that these volumes, initiated before C, Scheme, ML, Haskell, C++, Smalltalk, Eiffel, etc., are still readable and relevant. Knuth may not have picked up every criticism of earlier editions, but he has clearly read them and considered whether or not to adapt his text.

Over the decades, the scope of *Art* has waxed. We were told by Knuth early on that his plan encompassed several more volumes. Though there are still but three, the number promised has grown to volumes 4, *Combinatorial Algorithms*, 5, *Syntactical Algorithms*, 6, *The Theory of Languages*, and 7, *Compilers*. I hope Knuth’s plan is effected, mainly because I long to read volume 6.

But, as the titles reveal, as Knuth progresses, the topics become ever more specialized. However, I can recommend volume 1 to everyone: it is not a “quick read”; nor is it easy. It is a well-designed and well-written book on the nature of algorithms that are in daily use as we communicate with and through our computers.

Knuth invented TeX: these books are set in METAFONT, and a cursory tally reveals Arabic, Chinese, Devanagari, Japanese, and Korean entries in the index. An amazing piece of typesetting!

Even if you still own the first editions, you’ll want these for your library; if you don’t read them through, you’ll want them for reference.

P.S. For some reason that has not been revealed to me, Addison-Wesley (Longman) has let *Software Tools* go out of print (though *Software Tools in Pascal* appears to be available). Even if you’re not running RATFOR, Kernighan and Plauger produced a book that for 20 years embodied how programs could be made clean and easy to read, maintain and modify. It should be a must for any programmer not moving to Redmond, WA. C’mon, AWL!

### In the Picture

I’m an admitted fan of Icon. So I dove into *Graphics Programming in Icon* as soon as it arrived (thus). Because the high-level graphics features in Icon are integrated into the language (unlike, say, C and C++, which use additional graphics libraries), graphics code is briefer. The book is very fine. The color illustrations are informative (as well as lovely). The CD-ROM is useful, containing binaries for Windows and several UNIXes. What more can I say?

*Continued on page 76*
Keeping it Secret

Encryption, cryptography, and security have become buzzwords on the nation's business pages, but there are few good books on the topics involved. In fact, I'm disappointed. Though the two volumes I've just looked at are well-written and instructive, neither Loshin's Personal Encryption Clearly Explained nor Stallings's Cryptography and Network Security is without flaws. For me, the biggest of these is shared: neither author mentions ssh nor scp (rsh for scit). I learned about ssh last year from an article by John S. Quarterman (Matrix News, vol. 7, no. 2, Feb. 1997), and have found an increasing number of sites using it. I now use it in preference to Telnet. Tatu Ylonen and Timo Rinne in Finland deserve as much credit for this as Phil Zimmermann for PGP.

Loshin has another problem: pages 306-322 concern Outlook Express, and 322-324 concern Netscape Messenger. As most of you know, both of these are susceptible to wormlike viruses that attack the vulnerability of the overflow buffer. Tch, tch. Despite the excellent keynote last year by Butler Lampson (at the NT workshop), I think "Windows security" may be an oxymoron.

Loshin does do a good job on PGP and RSA SecurePC.

Stallings new edition is solid, but not complete.

And I'm still waiting for a complete book.

Digital Library

The British Library began a series of inquiries and projects concerning the "future" of the library in 1993. Several of the projects were part of the library's "Initiatives for Access Programme" and are presented in a lavishly produced book. (Carpenter, et. al, Towards the Digital Library)

A number of the essays affirm that the British Library will continue to conserve and acquire books and manuscripts. I hope so. But the general tenor of these repeated remarks is to render me pessimistic. And one wonders.

Will everything really be digitized eventually? We fret over losses of cinema, audio tape, etc. But we know and understand the limitations of paper — Cotton Vitellius A.x (Beowulf) in the library and the original Magna Carta bear ample witness to its survival power and that of parchment. But what do we know of CD-ROMs that makes us confident of their staying power?

Among the quotes from readers looking at the Sforza manuscript using the PIX program are several remarking that they could turn pages "just like a book" and see the illuminations as though they "had the book."

Really fascinating. Do we feel that these metaphors will die away? Or are the utilitarian marks on paper the standard to which all of our other methods of storage will be held?

In 1960, I obtained a "Manuscript Ticket" for the library, then in the British Museum. In 1967, finding the photographs unsatisfactory, I actually held the Beowulf manuscript in my hands and examined one page. Although the Sforza Book of Hours looks lovely on a good screen (and in this book's photographs), it does not look like the Book of Hours. I couldn't understand why for a while, but then I realized: it lacked scale. The small leaves of Beowulf and the elephant folios of Audubon are effectively equal.

We bring these works (in rendered form) to the public at large, but water them down through the democratization. ($19.95 prints of Van Gogh or Renoir or Cézanne are little like the paintings themselves; an acquaintance once complained to me in the Museum of Modern Art that she hadn't known that Dali's The Persistence of Memory was so small.

Some of the essays in Towards the Digital Library are extremely interesting. But what does it mean about the digital book that its references are shoddy and that there is no index?

Will I have to change my title to "The Digital Worm"?

Languages

Java network security? Some might claim that that is an oxymoron. After all, think about it. You download code from a source you may not know much about and then run it on your computer. With Java it is even worse than downloading a program or application; the download is done within the context of a Web browser, also running on your machine. So one view is that, to be safe, your only recourse is to disable Java in your browser so you know exactly what is being executed on your computer. After reading this book you might feel more comfortable with some less stringent measures.

Java Network Security provides a good overview of the issues involved in maintaining security on a computer or network running Java. This book does not provide an overview of the Java environment itself, so there is an assumption you are already familiar with at least the higher level features of that environment.

Beyond some superficial distractions, which I'll talk about later, this book takes the right approach to these issues. There is a danger in writing books about technology, especially a volatile new technology such as Java. The Java specification has had many changes in a short period of time, and many major variations (such as Java Beans) are introduced frequently. So in a sense any book is dated by the time it reaches a printing press. Java Network Security is not a "how-to" book. There are few specific guidelines on how to implement security in the Java environment. Rather, much of the book discusses the issues and problems and how they apply to the Java environment.

For example, consider the class loader, which is responsible for taking the source information for a Java class and loading it into the Java Virtual Machine for use by other Java applications or applets. The authors do not give instructions for building your own loader that avoid the problems. In fact they even point out that such instructions are beyond the scope of the book. So what do they write about for seven pages?

They provide a detailed description of the requirements for a class loader, the process involved in loading a class, considerations for building your own, and finally motivation for why you might want to build your own. For each the authors describe the differing requirements or situations for loading a class from a local source or from a Web server. They also discuss the need for a trusted core of code in the Java environment. Basically, you need this trusted core to ensure a safe and sound starting point for building other applications or applets.

With the pace of technological change, these detailed descriptions are more valuable than any set of how-to instructions. They give administrators or developers the in-depth understanding of what is going on so they can make their own decisions about what measures are needed for their individual network or application. Even better, much of the discussion in the book is geared toward issues of security analysis — specifically, which threats each countermeasure might be appropriate for. So the book will be useful to a Java developer who is not familiar with some of the security issues as well as to the security engineer or network administrator trying to ensure any work done in Java still complies with existing security requirements and policies.

Unfortunately, many readers might not get that far. The book often assumes a condescending tone about Java in general. The authors do acknowledge some of the already known security problems with Java, but tend to downplay them. In many cases they acknowledge security holes in Java that exist in other technologies, but then make vague statements as to why the holes are less of a problem in Java. They often rely on the "Web of Trust" resulting from signed certificates to build up the apparent strength of Java. Still, often this Web sounds more like a leap of faith that you should trust others to provide sound, well-designed, and well-built applets. The problem, of course, is implementation errors, even in signed and registered applets.

Security is provided both by those who build the applets and those who maintain the networks. Yet the book puts the greater responsibility on those who administer the networks. "So in the absence of implementation errors, either on the part of the browser vendors or on the part of the computer operators, administrators and system programmers, Java should be safe" (emphasis in the original). The paragraph goes on to say that the browser vendors have a good reputation for dealing with these things and the book is intended for the administrators.

This patronizing attitude is most prevalent in the first chapter. If you can make it through this material you find the gems in the remainder of the book.

You won't find many ready-made solutions, but you will find a wealth of technical information that will help both network administrators and developers understand the Java environment. The book includes a CD with a Java development environment package and some other tools, and it also has the source for the various examples in the book. So you can try things out for yourself.

This book will be of value to anyone interested in technical details behind Java. It will be of obvious interest to administrators and developers. Yet others might be interested since many of the issues described apply to other networked applications. But it is to those working with Java that I most highly recommend this book.
Adrian Cockcroft with Richard Pettit

Sun Performance and Tuning: Java and the Internet


Reviewed by Andrew Hume
<andrew@research.att.com>

If you own or run a Sun system, then you must get this book. Nitpicking aside, there is such a wealth of information and experience presented here that it would be folly not to read this book. I'd prefer it wasn't so necessary to have this book to make my Sun perform well. I'd prefer that more than a few of the Sun support hotline troops had read it. Never mind; just get the book.

The book covers many areas; chapter 1 is a quick summary of tips and hints for those too impatient to read the whole book. Chapters 2 and 3 cover the general principles of performance management and measurement, including reviews of various commercial products. Chapters 4 and 5 cover "the Internet," or at least how to deal with http servers and Java. Chapter 6 covers the way too complicated area of how to optimize code, particularly with respect to instruction sets and architectures.

Chapter 7 covers high-level application tuning, that is, how an application deals with Solaris, such as tracing and file systems. Chapter 8 is a long and involved discussion of disks, RAID boxes, and controllers. Chapter 9 covers networking, and chapter 10 talks about processor analysis (mutexes, memory, CPU caches). Chapter 11 is a gory description of various Sun architectures, down to memory, I/O bus, and backplane issues. Chapter 12 talks about various system caches (generic, filesystem, and networking). Chapter 13 details how Solaris uses your RAM and swap space.

Chapter 14 covers many of the kernel algorithms pertinent to, or controllable by, the user and how to measure and tune them. Chapter 15 covers, sometimes in exasperating detail, the techniques for getting useful metrics out of Solaris. Chapters 16 and 17 cover the SymBel language and environment, which is a domain-specific language aimed at performance monitoring and analysis. Appendix A is a terse summary of the useful kernel tunables, and appendix B is a compendium of useful references and resources. An index rounds out the book, which totals 587 pages.

In general, this book is a good, solid read; the information is accurate and well presented. The various sections on disks, in particular, are excellent, and the detailed example (pp. 186-194) of figuring out a small disk I/O problem is outstanding. The discussions of the various CPU and machine architectures, and how to optimize applications for them, is also very good. Finally, the se system is just a good idea, and I appreciated that all the scripts described in the book come with the se software and can be tried as you go.

It is rather a pity that such a good book has a few small problems. As I nitpick, if I seem a tad testy, it's only because I have been having a tuning battle with Solaris for several months.

The book is not well proofread. I picked five pages at random and found typos or other errors on two of them (for example, the contents line [page xv] for page 432). There is too much Sun official line for my taste. (How many times do we need to be told Sun bought Encore? And that Encore's RAID systems are really good? I would guess once, not three or four times.) And the gratuitous "official position" on dynamic libraries is very defensive in tone and doesn't address the real problems that it brings to applications. And the constant toutting of Sun products wears thin really quickly. (If the RSM2000 RAID box is that good, why is there still a problem on reboots with Solaris not recognizing all the RSM2000 devices? The stock reply of "just one more reboot -r" sounds like it might have come from Redmond.)

All the maxims presented seem true, but several have important caveats missing. For example, on page 171, we have "It is pointless putting large amounts of memory inside a hardware RAID controller and trying to cache reads with it." Ordinarily, this is true; the disk buffer cache eliminates these references. Except, as Cockcroft mentions later on, if your performance requirements mandate using the disk as a raw or direct I/O device, then the buffer cache is not involved.

There is a fair amount of chest pounding on Sun performance, especially disk and file system. I simply don't believe any of it (although I'd like to). Claims are presented without any setup or measurement details. For example, on page 215, Cockcroft asserts the RSM2000 RAID can sustain 66MB/s. I've been trying to do this for a few months - the best I can get is 45MB/s. I'd love to know what the details are behind the 66MB/s (for example, reading or writing? striped? RAID? block size? kernel parameters?). And as for reading any file at 20-30MB/s (page 181), you will run into problems with any filesystem, not just UFS. It would really be helpful here if Cockcroft could actually give real details on how you might accomplish such speed. And although the StorageTek Redwood drives are fast, they are not 15MB/s (page 184). Their rated speed is about 11MB/s; on my system, we occasionally see 10MB/s or so.

Cockcroft exercised one of my pet peeves when he discusses *time* on page 158; it provides "accurate and high-resolution process timing." He gives an example of comparing two timings: one from long ago where (apparently) *usr*+*sys* was .6 seconds, and a recent one where *usr*+*sys* was .014 seconds, and concludes a speedup of 43. This is such sloppy science, Cockcroft must have had a brain...
cloud when he wrote this. Ptime's precision is apparently 0.5 ms (although, charmingly, this is not stated on the manual page), but the accuracy is rather less than this. As an example, I ran the same command with Ptime three times on my Sun E10000; the usr+sys were 28.2+14.5, 31.4+18.3, and 33.6+16.7. (I rounded up to the nearest .1s.) Here the accuracy is about ±10%, or 3s. So much for millisecond precision.

Finally, I regret Cockcroft didn't say more about variables I have found to be fairly important. The tunable maxphys specifies the maximum size of a physical I/O to disk; its default value is 128KB, which is way too low for many disks. The xca1 column from mpstat represents, we are told, the number of cross-processor interrupts per second; what are good/bad values here? The answer probably varies by configuration, but I bet there is a heuristic, as there is for many other values (such as the scan rate sc).

Despite these problems, let me emphasize that I liked this book, and regard it as mandatory for any serious Sun user or administrator.

Timothy A. Howes and Mark C. Smith


**Reviewed by George W. Leach**
<gwleach@gt.net>

The Lightweight Directory Access Protocol (LDAP) has emerged over the past couple of years as a key enabling technology for building large-scale distributed systems. Originally devised as a lightweight alternative to ISO technologies for client access to X.500 directory services, LDAP has evolved to become a universal access protocol for accessing both standards-based (X.500, NIS, DNS, DHCP, DCE) and proprietary directories (Microsoft Exchange, Novell NDS, Microsoft Active Directory in NT 5.0, Lotus Domino, Lotus C/Mail). And LDAP standardization efforts have evolved to encompass support for native LDAP servers as well.

**LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol** is the only book published on this important topic to date. The authors' names are affiliated with just about every important RFC associated with LDAP and with good reason. They were two of the key people responsible for the development of the original LDAP specification and implementation at the University of Michigan. Both have since moved on to help define the next version of LDAP (v3) and develop a commercial implementation of an LDAP server at Netscape Communications.

The focus of this book is on the C Programming Language Application Programming Interface (API) for client programs accessing directory services via LDAP. Two different C API Software Development Kits (SDKs) are covered in this book—the University of Michigan and Netscape SDKs. Both versions described in the book correspond with v2 of LDAP and are now superceded by the emerging v3 specifications. The authors point out differences between the two APIs, which are relatively few, throughout the book.

A minimal overview of LDAP, its capabilities, and how the schema of an LDAP server is designed is provided in the first three chapters. The remainder of the book dives into a detailed look at the C API for invoking LDAP capabilities to make inquires or updates upon the server. The organization of introducing new API calls flows in a logical manner and is supported with numerous examples and code listings. There is a great deal of repetition in the book. Variations are made on example code, which can get quite lengthy at times. Many API calls take some of the same parameters and provide some of the return types. Each new API call provides a complete listing and description of each parameter and return type. But this is fine because it makes it more usable as a reference.

Although it may get rather tedious to read at times, this book does give the reader a very comfortable feel for the capabilities of the API and hence LDAP from a client perspective. And despite the fact that the book is now somewhat out of date with the evolution of LDAP and the C API, the backward compatibility with v2 still make this a worthwhile book to read.

Both SDKs are freely available on the Internet. The University of Michigan implementations of both the C API and the LDAP server have not been enhanced since the spring of 1996 and support only v2. The Netscape C API SDK and Netscape Directory Server have been enhanced to support the proposed v3 LDAP capabilities and remain backward compatible with v2.


The version of the C API corresponding to this Internet Draft is available from Netscape: [http://developer.netscape.com/tech/directory/downloads.html](http://developer.netscape.com/tech/directory/downloads.html).

More information on LDAP may be found at the following URLs:

- [http://search.netscape.com/newrel/ref/ldap.html#server](http://search.netscape.com/newrel/ref/ldap.html#server)
Managing Mailing Lists


Reviewed by Rick Umali
<rgu@world.std.com>

This book is a fine start for aspiring list managers and system administrators on UNIX systems, but intermediate to advanced mailing list managers may expect a little more.

Given the title, I was expecting more on how to "manage" a mailing list, not so much the technical issues, but rather the nontechnical issues (unruly behavior, policies, etc.). There are preliminary discussions on mailing list management topics, but I was hoping for more.

Instead, the primary focus of the book is how to start and manage a mailing list manager (MLM). The author writes for both the list's maintainer/manager and the list's server administrator as well. This is an important distinction. One person runs the list, sometimes moderating the mail messages of the list's subscribers, while the other manages the software that manages the list. Although Schwartz does cater to both audiences, the scale tips slightly to the "server administrators" as the primary audience.

Despite this focus, the book has plenty to offer for both groups.

Because I run a small mailing list (the Tiger Woods mailing list) using Majordomo, a lot of the general concepts (such as how email works) in the first two chapters were familiar to me.

Schwartz's treatment of these general concepts is good.

You learn about the mysterious ">") before the word From if From is the first word on a line in the body of an email message. You also learn about Message-IDs and the difference between MUA (mail user agent) and MTA (mail transfer agent). I was especially pleased to learn about the RFC 1893, which specifies codes returned by mail bounces. And Mr. Schwartz compares mailing lists to USENET News, as well as how to handle large mailing lists.

Table 2-1 in chapter 2 is a good Consumer Reports breakdown of the different features provided by the four mailing list managers described in the book. Not every reader will have the flexibility to choose which list management software to use, but Mr. Schwartz provides advice on that as well (choose a server that runs either LISTSERV Lite or Majordomo, because list configuration can be done by the list manager in addition to the server administrator).

The next four chapters touch upon creating/configuring the list, writing the "welcome" message for your list, testing the list, and then list administration. Each chapter attempts to describe the details of the specific MLM. The author provides good details on the Majordomo and LISTSERV Lite, primarily because these MLMs allow the list manager (in addition to the server administrator) to configure the list.

Chapter 7 ("Maintaining Lists with Sendmail") may be old hat to seasoned system administrators, but it's a great introduction to how some companies set up internal mailing lists. This is the most raw chapter and the most daunting for the beginner, although budding sysadmins will definitely want to try this out.

Chapter 8 ("Troubleshooting Your Lists") is short, but the discussion of how to debug mailing list managers is important for server administrators. Mr. Schwartz's own example of how he resolved a mailing list loop ("A Third-Party Loop," p. 101) provides great insight into solving mailing list problems.

The last four chapters on administering Listproc, Majordomo, SmartList, and LISTSERV Lite are geared for the server administrator. This person may be the system administrator. Each chapter provides how to compile/install each kit, how to configure the "server," how to add lists, and how to deal with the day-to-day management of the list software itself.

Each chapter is like an expert looking over your shoulder as you obtain the MLM software and set it up. List managers should peruse the chapter concerning their MLM packages, because it will provide insights on what can be done with the mailing list. In these chapters, I learned (among other things) that Listproc and LISTSERV Lite employ servers, how Majordomo generates the "bounced email list," and that SmartList uses procmail (a book in and of itself).

Chapter 8, on troubleshooting your list, and chapter 2 represent the content I was most looking for. There will always be more list managers than server administrators because MLMs are designed to support many lists. Because of this, I was hoping to see more information on how to deal with list management issues such as irate list subscribers, mail list bombing, and how to "grow your list." In other words, the "diplomacy" side of mailing list management.

Mr. Schwartz does touch on this side. His comments and examples on how to write an effective Welcome Email should be standard reading for new list managers.

Chapter 2's essay on "The Life Cycle of a List" (p. 22) (submitted by Brent Chapman) is an excellent "time line" on the evolution of a mailing list (from creation, to growth, to either stagnation or maturity). I was hoping for more of this, because the setup and configuration are typically done once, or infrequently, compared with the day-to-day grind of maintaining a list. For this sort of thing, the list managers's mailing list (which the author mentions in the preface) will prove to be an excellent supplement to Managing Mailing Lists.
I attended less of the LISA-NT conference than I would have liked, but what I saw was interesting. USENIX is most grateful to Remy Evard, Ian Reddy, and their program committee for all the hard work in developing the program. Despite Microsoft's goal of zero system administration, I predict a lot of employment in managing PCs in the foreseeable future. New features in NT 5.0, such as the inbuilt support for HSM (hierarchical storage management), will confuse users and require a bunch of system configuration decisions by sysadmins. Growing interest in high availability computing and security will also require far more attention in the short term.

The SAGE executive committee meeting went well. As usual, now that the committee is working well together, it is faced with elections later this year.

All in all, I had a great time, as did nearly all the attendees I spoke to. A substantial part of this success is due to our primary liaisons at Microsoft: Jim Gray, Mike Jones, and Todd Needham. They have been instrumental in getting us the right speakers giving the right talks from Microsoft, and providing access to the real developers (mostly through BoFs). Together with the program chairs they manage to do all this without the audience feeling like they were "Microsofted." Finally, I'd like to thank the entire USENIX staff for their hard work in organizing, promoting, doing the proceedings, onsite logistics, and everything else that is involved in making our conferences so successful. Judy DesHarnais and her staff and the hotel did an excellent job on site (despite some problems due to changing reservation systems); I was particularly impressed with the A/V guy.

I look forward to returning to Seattle next year; hope to see you there!
Summary of USENIX Board Meetings

by Ellie Young
Executive Director

<ellie@usenix.org>

Here is a summary of the actions taken at the regular meetings of the USENIX Board of Directors held in March and June, 1998.

New Affiliate Member Category

The Europe.n.ES (the Swedish National Group of EurOpen), with 300 members, had accepted a proposal from USENIX to offer its members all individual member benefits, except voting privileges. Groups approved for affiliate memberships must have a minimum of 100 affiliate members who will share a common expiration date. Any member of an approved group is eligible to become an affiliate member. The group will make a single payment annually to USENIX for their affiliate members. The designated representative of any group will be eligible to be an individual member.

Institutional Member Benefits

Institutional members will be offered, later this Fall, a new menu of benefits to choose from, as follows:

- subscription to the printed proceedings and access to the online library for all patrons or department members. (Educational dues: $400; corporate dues: $700)
- access to the online library for all patrons (if a library or departmental students/faculty (if an academic department). (Educational dues: $300; corporate dues: $500)
- subscription to the printed proceedings only (educational dues: $175; corporate dues: $325)

NLnet Foundation

Honeyman was appointed the liaison to review proposals and offer guidance on projects they might be funding to support electronic information transfer.

Finances

It was agreed that Geer, Hume, the CPA, and Young would draft an investment policy and make a recommendation about the future management of the fund.

Computing Research Association

In order to increase our visibility to the academic CS community, USENIX sponsored a booth at the CRA’s bi-annual conference in Snowbird in July.

International Speaker’s Bureau

It was agreed to allocate funds and establish a “USENIX International Speaker Program” for other similar organizations who would like help in finding speakers. The program will pay transportation and some other expenses for speakers to suitable events.

Sponsorship of Other Conferences

It was decided that while we could not fully sponsor a workshop on agent systems and programming, USENIX could be a co-sponsor with the IEEE and help with planning, promotion, and proceedings.

It was agreed to have “in cooperation” sponsorship (providing help with promotion and publicity) of the ACM/IEEE MobiCom conference and the 2nd IEEE Workshop on Mobile Computing Systems and Applications.

Based on the SAGE Executive committee’s recommendation that SAGE contin-
ue to be a co-sponsor of the SANS System Administration Conference, an agreement between USENIX and SANS Institute for continuing cooperation for the next two years was approved.

**USENIX Conferences**

*Intrusion Detection*. A proposal from Marcus Ranum and Dan Geer to sponsor a workshop on this topic was accepted. Geer will serve as board liaison.

*Nordic EuroOpen/USENIX Conference*. The Swedish affiliate representative's request that USENIX support a conference in Stockholm in February 1999 was approved.

*Network Administration Conference*. SAGE's request that they co-sponsor a third conference in 1999 along with a proposal from Williamson and McDonald on this topic were approved.

**USENIX Annual Conference 1999**. A proposal from Avi Rubin to serve as program chair was accepted. Jordan Hubbard will serve as chair of the freenix track. Honeyman will serve as board liaison and on the program committee. It was also decided to expand the tutorial program to 3 days.

*Win/NT and LISA-NT conferences*. It was agreed that for the next few years these events should be held in Seattle and co-located.

*Security Symposium*. After polling the recent and past organizers, it was decided that we will repeat this symposium every 12 months after August 1999.

*Smart cards*. A proposal from Scott Guthrey and Peter Honeyman to sponsor a research-oriented workshop co-located with the CardTech/SecureTech conference in May 1999 was approved.

**Committees/Liaisons**

At a short meeting of the newly elected Board in June, the following appointments were made:

Andrew Hume will serve as the board liaison to the Nordic, COOTS, Windows/NT and Domain Specific Languages conferences. Jon “maddog” Hall will serve as liaison to SANS. Hal Pomeranz was appointed as the USENIX board representative to the SAGE executive committee. The following committees were formed and/or new members were appointed to fill the posts held by outgoing board members:

- Executive: Hume, Geer, Pomeranz, Young
- :login: editorial: Darmohray, Zwicky, Kolstad, Farrow, Cohen, Hénon, Young, and Pomeranz

- Prizes & Awards: Hume (chair), Hall, Seltzer, Groundwater, O'Dell, Ritchie, and Young
- Selection of LISA '99 Program Chair: Parseghian, Rose, Young (with Kreiling and Harrison from SAGE)
- Scholastic: Honeyman (chair), David Kotz, Darrell Long, Parseghian, Young, Barnett
- STG Committee: Hume, Geer, Zwicky, Young
- Tutorial Review: Klein, Honeyman, Geer, Seltzer, Rik Farrow, Brent Welch, Hall, Avi Rubin, Young.
A large group effort headed by Warren Toomey has undertaken to preserve the very old versions of UNIX that ran on the PDP-11s. This group, under the name “The PDP-11 UNIX Preservation Society,” or PUPS for short, has managed to convince the Santa Cruz Operation (the current owners of UNIX) to issue cheap personal-use UNIX source code licenses for research editions 1 to 7 and 32V.

Having 32V licenses available has opened the door for the re-release of the BSD versions of UNIX from the Computer Science Research Group at the University of California, Berkeley. You need only a 32V source license to get the full source code of all the BSDs, including 4.4BSD.

Of course, you can obtain 4.4BSD-Lite without requiring a source license.

Kirk McKusick has now released a 4-CD set that contains the source code for all of the CSRG work. This includes:

- 1BSD, 2BSD, 2.79, 8.9, 10 BSD, 3BSD
- 4.0, 4.1a, 4c, 2 BSD
- 4.3 [reno, tahoe]-BSD
- The Net/1 and Net/2 releases
- 4.4BSD and 4.4BSD-Lite2
- complete SCCS logs of the CSRG development

Information on obtaining a personal source license and ordering this four-CD set can be obtained from:

<http://www.mckusick.com/csrg/>

This site also has links to the PUPS home page and related sites that are interested in older versions of UNIX.

There was a short September 1978 ;login; and then the next issue did not appear until January 1980. In that issue it says that the Association is once again functioning, that it was decided to omit the July-December 1979 issues and publish the other missed issues “shortly.” Tom Ferrin reported to me, “These were in fact never published, and the next issue was the August 1980 issue, in which it was announced that there was a new newsletter editor. The next issue after that was January 1981, and the issues seem to flow regularly thereafter.”

I asked Lou Katz, then the President, about the hiatus, he told me; “I don’t quite remember who all the guilty players were. I think (but am not sure) that Mel began to fall further and further behind (he was moving from Brooklyn to Rockefeller and was seeing through the incorporation), and maybe someone was supposed to pick up the thread. Maybe I said I would – I don’t think so, but it is possible.” The Association did not cease functioning, but perhaps there was a gap.

Mel Ferentz, who had been running the newsletter from a corner of his desk, reported, “After I went to Rockefeller (1978) things got pretty chaotic. The Rockefeller lawyers suddenly got very concerned about how USENIX was being run. They insisted we be incorporated and in their white-shoed-way decided we were a ‘trade association’ and could not be tax exempt. That kept me busy for quite a while with lawyers, banks, etc.

“I remember a Board meeting at which Wally [Wedel] attacked me and at which the board decided that he would take over the future of the newsletter and I would attempt to catch up with the past. (Neither of us followed through.) [My guess is that this was the June 1980 meeting at the University of Delaware.]

“I don’t remember when the Austin meeting was [June 24-26, 1981]. There was a threatened airline strike, and, good union man that I am, it was the first meeting I didn’t attend. That was the meeting Wally was doing the Proceedings, which also never appeared.

“By the election of 1981-1982, I left the board and Tom [Ferrin] took over as treasurer.”

There was actually even more that was taking up the time and energy of Lou, Mel and the rest of the volunteer board (with no paid staff): meeting planning (October 1978 at SRI in Menlo Park, January 1979 at ISC in Santa Monica, and June 1979 in Toronto). And, as can be seen from these locations (and the addition of Debbie Scherrer and Tom Ferrin to the board), there was a geographical shift from the Northeast to California. (In 1981, Lou Katz was to leave Columbia for Berkeley.)

Tom Ferrin deserves the final word, “The long and short of it is that the group was going through growing pains as it matured from a rag-tag group of volunteers into a bonafide Association with a board of directors, albeit all of whom were still volunteers.”

With no ;login; to cover, I will talk about the Santa Monica meeting next issue.
USENIX Annual Awards

At the annual USENIX Technical Conference held in New Orleans in June, Andrew Hume announced the winners of two annual awards.

Tim Berners-Lee received USENIX’s 1998 Lifetime Achievement Award, which recognizes and celebrates singular contributions to the UNIX community in both intellectual achievement and unparalleled service. He was honored for “spinning the Web that has helped to transform the Internet into a fundamental part of everyday life and for his continued evangelism on its behalf,” as inscribed on the original glass sculpture. Tim invented the World Wide Web in 1990 while at CERN in Geneva, Switzerland. He wrote the first WWW client and the first WWW server, along with communications software defining URLs, HTTP, and HTML.

The Software Tools Users Group (STUG) 1998 Annual Award recipient was John Ousterhout for Tcl/Tk, the software tools for which he is best known. Together or separately, Tcl/Tk are much used, and they exhibit the spirit that STUG was founded to encourage: portability, adaptability to seemingly unfriendly environments, and clarity of concept.

Torn Money

Stay tuned to the USENIX Web site, and read the December 1998 issue of login, to learn about what is being done as a replacement for the PGP Key Signing Service – Torn Money!

USACO Camp Completed

by Rob Kolstad

Rob Kolstad, editor of login, is head coach of the USA Computing Olympiad Team
<kolstad@usenix.org>

Fifteen of the USA’s brightest young programmers joined five coaches in Racine, Wisconsin, June 30-July 8, 1998 for the annual USA Computing Olympiad Training Camp. This USACO-sponsored event brings together the best and brightest to choose the team of four that represents the USA in the next International Programming Contest, which will be held September 4-11, 1998, in Setubal, Portugal.

After 8 days of training and two grueling five-hour coding contests, the four team members were chosen. They are:

- Third time international competitor Matt Craighed, a Senior at St. Paul Academy in St. Paul, MN. Matt plans to enter MIT in the Fall.
- Adrian Sox, a Senior at Upper Dublin H.S. in Fort Washington, PA. Adrian represented the USA last summer in Poland and plans to enter Carnegie Mellon in the Fall.
- Alex Wissner-Gross, a Junior at Great Neck South H.S. in Great Neck, NY.
- Chuong Do, a Sophomore at Garland H.S., Garland, TX.

The 1998 USACO finalists were selected by considering their rankings and performance in the three Internet competitions (Fall, Winter, and Spring), and the recent National Championship, along with the ability to attend the summer camp, June 30-July 8, 1998.

Here’s the easiest question from the second five-hour competition in Wisconsin: Friendly Coins (by Dan Adkins, USACO 1995-1997)

Most farmers know that cows are mathematically challenged. They make for poor workers in fast food restaurants, since they have trouble making change for a dollar. Farmer John has decided to ensure that the cows are employed making change only for those currencies in which the
cows can use the “give highest coin” principle to make change for any possible amount of change. These currencies have “friendly” coins.

Consider the coin set \{1, 2, 5, 10\} and make change of 32 cents using the “give highest coin” principle. This principle would yield three 10-cent coins, and one 2-cent coin, which is four coins and the minimum number of coins for that value of change. In fact, for all values of change, this coin set will require only the minimum number of coins when giving change by the “give highest coin” principle. Thus, it is a friendly coin set.

On the other hand, the coin set \{1, 4, 5\} is unfriendly because if you are giving 8 cents change, you will give one 5-cent piece and three 1-cent pieces, a total of four coins. However, it would be better to give two 4-cent coins. An unfriendly coin set is one for which the “give highest coin” principle does not give the smallest number of coins for one or more change values.

You must help Farmer John determine whether coin sets are friendly or not.

**Input (file INPUT.TXT)**
The first line of the input contains \(N\), the number of different coins available \((1 \leq N \leq 20)\). The next \(N\) lines give the coin values, one per line. The coin values will be between 1 and 15,377 inclusive.

**Sample Input**
3
1
4
5

**Output (file OUTPUT.TXT)**
If the coins are friendly, the output on the string “FRIENDLY.” Otherwise, output a value where the greedy algorithm does not work. If a counter-example exists, it will not exceed 1,084,673.

**Sample Output**
8

---

**1998 USACO Finalists**
Congratulations to these fine programmers!

<table>
<thead>
<tr>
<th>Grade</th>
<th>Name</th>
<th>State</th>
<th>School</th>
<th>Teacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Bao, Joshua</td>
<td>PA</td>
<td>State College Area School</td>
<td>Mary Bytheway</td>
</tr>
<tr>
<td>10</td>
<td>Cheng, David</td>
<td>DE</td>
<td>Brandywine HS</td>
<td>Mr. Shea</td>
</tr>
<tr>
<td>12</td>
<td>Connell, Brendan</td>
<td>MD</td>
<td>Montgomery Blair</td>
<td>Karen Collins</td>
</tr>
<tr>
<td>12</td>
<td>Craighead, Matt</td>
<td>MN</td>
<td>St Paul Academy</td>
<td>Bill Boulger</td>
</tr>
<tr>
<td>10</td>
<td>Danaher, John</td>
<td>VA</td>
<td>TJHSST</td>
<td>Sally Bellacqua</td>
</tr>
<tr>
<td>10</td>
<td>Do, Chuong</td>
<td>TX</td>
<td>Garland HS</td>
<td>Linda Donahue</td>
</tr>
<tr>
<td>11</td>
<td>Haspel, Daniel</td>
<td>VA</td>
<td>TJHSST</td>
<td>Sally Bellacqua</td>
</tr>
<tr>
<td>11</td>
<td>Impollonia, Bobby</td>
<td>NY</td>
<td>Stuyvesant HS</td>
<td>Mike Zamansky</td>
</tr>
<tr>
<td>12</td>
<td>Knock, Bobby</td>
<td>TX</td>
<td>Langhan Creek HS</td>
<td>Scotty Johnson</td>
</tr>
<tr>
<td>12</td>
<td>Liang, Percy</td>
<td>AZ</td>
<td>Mountain Pointe HS</td>
<td>Jon Gompert</td>
</tr>
<tr>
<td>12</td>
<td>Maclsaac, Sean</td>
<td>VA</td>
<td>TJHSST</td>
<td>Sally Bellacqua</td>
</tr>
<tr>
<td>12</td>
<td>Mellis, David A.</td>
<td>IL</td>
<td>IMSA</td>
<td>Ron Vavrinek</td>
</tr>
<tr>
<td>12</td>
<td>Sox, Adrian</td>
<td>PA</td>
<td>Upper Dublin HS</td>
<td>Dr. Sharon Traver</td>
</tr>
<tr>
<td>11</td>
<td>Stanek, Sean</td>
<td>IA</td>
<td>Central Academy in Des Moines</td>
<td>Scott Schoneberg</td>
</tr>
<tr>
<td>11</td>
<td>Wissner-Gross, Alex</td>
<td>NY</td>
<td>Great Neck South HS</td>
<td>Albert Cavallaro</td>
</tr>
</tbody>
</table>

**USACO – Letters from Students**

[Below are letters we received from high school students who participated in the USACO training camp.]

Dear Ellie Young,

I want to thank USENIX for providing me with the opportunity to attend the USA Computer Olympiad training camp this summer. I appreciate the generosity of USENIX, and am grateful to have had this opportunity. I had a wonderful time in Wisconsin, and the camp was a great learning experience. I was able to refine my computer programming skills significantly. I particularly liked the lectures and labs.

As a result of this experience, I was selected to represent the United States at the International Olympiad in Informatics in Portugal, for which I thank you again. I am extremely excited about this opportunity, to which I have aspired for three years. Thank you for making my dream a reality.

Gratefully yours,

Alex Wissner-Gross,
Great Neck South High School, NY
xanzak@idt.net

Hello!

My name is Chuong Do, a participant at the recent USA Computing Olympiad training camp. I just wanted to write to thank you for sponsoring such a great program; the camp experience was not only enjoyable, but I was truly able to learn quite a bit about proper techniques...
for problem solving and algorithm design. I appreciate your support for such a wonderful contest. As I have just finished my sophomore year, I am looking forward to being a part of the USACO for a few more years in hopes that I can give back some of what it has done for me.

Ever grateful,

Tom Do

Hello.
I was just a finalist at the USACO training camp and would like to thank your organization for sponsoring such a great program. I even heard that you (as a group) are going to host the IOI in the United States in 2003, which is great, even though I will have graduated by then. Anyway, I just wanted to say that your help is greatly appreciated and felt by all of us. Thanks for a great computing olympiad!

David Cheng

Thank you SO much for sponsoring the USA Computing Olympiad and all the activities! It would be impossible to do it without you. Thanks again!

Sean Stanek
USACO Finalist

Analogue photography by Snoopy of the LISA costume party last October. The concept for the party was to come dressed as your favorite system command, computing equipment, or high-tech concept.

A. string B. malloc C. cd D. chmod E. Garage Sale F. mask G. top wrappers H. box
Tutorial Program  Sunday-Tuesday, December 6-8, 1998
36 tutorials covering: performance tuning, heterogeneous systems management, Windows NT administration, TCP/IP administration, networking, sendmail, Solaris administration, security, DNS, Ethernet, Perl, Samba, PC/UNIX connectivity, distributed systems, and more.
Register now to guarantee your first choice—seating is limited.

Technical Sessions  Wednesday-Friday, December 9-11, 1998

Wednesday, December 9, 1998

Joint Opening Session
Opening Remarks & Awards  Xev Gittler and Rob Kolstad, Program Co-Chairs
Keynote Address  Eric Allman, CTO, Sendmail, Inc.
Eric Allman is the original author of sendmail. He was an early contributor to the UNIX effort at UC Berkeley, authoring syslog, tset, the -ma troff macros, and trek. He was the chief programmer on the INGRES database management project and designed database user and application interfaces at Britton Lee (later Sharebase), and contributed to the Ring Array Processor project for neural-network-based speech recognition at the International Computer Science Institute.
Eric is the CTO of Sendmail, Inc., and gives tutorials and presentations at USENIX conferences.

Refereed Papers

Security
Titan
Dan Farmer, Earththink Network; Brad Powell,
Sun Microsystems, Inc.; and Matthew Archibald,
KLA-Tencor Corporation
Infrastructure: A Prerequisite for Effective
Security
Bill Fitzen and Jeff Carpenter, CERT
Coordination Center, Software Engineering
Institute, Carnegie Mellon University
SSH: Extending SSh for Secure Root
Administration
Christopher Thorpe, Yahoo!, Inc.

Invited Talks

Panel Discussion: TCP/IP Futures
Moderator: Phil Scarr, Global Networking and
Computing, Inc.
Is IPv6 really coming? What about IPSEC? And why
do we need something new, anyway—what’s wrong
with IPv4? Our panel will discuss these topics, and
muse about the future of networking.
Panelists: To Be Announced

Practicum

Teaching System Administration
How does our profession develop new administra-
tors? Are universities the answer? Extension pro-
grams? In-house training? This session will touch on
the most important topics of educating system
administrators.

Practicum

FREENIX Administration Issues
The proliferation of free UNIX-clones like Linux,
FreeBSD, and NetBSD offers a new set of both
technical and political challenges. This session dis-
cusses those challenges and more.

Freeware

University Issues
Universities are a special environment with tech-
nical and political challenges all their own. This ses-
sion will discuss some of those issues, such as net-
working, mail/dialup solutions, security, civil rights,
and help desks.

Pushing Users and Scripts Around
System Management With NoteScript
Aparatim Purakayastha and Ajay Mohindra,
I.B.M. Thomas J. Watson Research Center
AccountWorks: Users Create Accounts on SQL,
Notes, NT and UNIX
Bob Arnold, Sybase, Inc.
Single Sign-On and the System Administrator
Michael Grubb and Bob Carter, Duke University

Zero to LISA in One Year
Brent Chapman and Paul Evans, Covad
Communications Company
How do you establish, provide, and scale system and
network support for a company whose employee
count doubles every four months, and whose site
count doubles every 6 months? Come along for the
ride with a hyper-growth startup as we explain how
to cope with growing from 50 people at 2 sites in 1
region to 400 people at 8 sites in 6 regions in less
than a year.

Storage Performance
How We Backed up Our 6TB Sun E10000
W. Curtis Preston, Collective Technologies; and
Rob Cotter, Hughes Space and Communications
Configuring Database Servers
Christopher R. Page, Millennium
Pharmaceuticals
General Rules for Maximizing Disk I/O
Performance
Dan Pollack, America Online

Got LDAP? Deploying and Using the Lightweight
Directory Access Protocol
Leif Hedstrom, Netscape Communications
Corporation
Deploying and managing a directory server is a com-
plicated task, requiring serious planning, a good
architecture, and an idea of what to achieve. This
presentation will introduce LDAP to the audience,
give an outline of how to deploy LDAP, and present
some possible solutions to deploying LDAP. We’ll
also talk about how to decide on software and hard-
ware architecture, making sure you know how to
select the appropriate tools for your environment.
Thursday, December 10

Refereed Papers

Distributed Computing
A Configuration Distribution System for Heterogeneous Networks
Glédson Elias da Silva and Fabio Q. B. da Silva, Federal University of Pernambuco
An NFS Configuration and Management System and its Underlying Object-Oriented Model
Danielle Franklin, Fabio Q. B. da Silva, Juliana Silva da Cunha, Luciana Varejão, and Rosalie Belian, Federal University of Pernambuco
Design and Implementation of an Administration System for Distributed Web Server
C. S. Yang and Myung V. Luo, Institute of Computer and Information Engineering, National Sun Yat-Sen University

Networking
MRTG, Multi Router Traffic Grapher
Tobias Oetiker, Swiss Federal Institute of Technology
Wide Area Network Ecology
Jon T. Meek, Edwin S. Eichert, and Kim Takayama, American Cyanamid Company
Automatically Selecting a Close Mirror Based on Network Topology
Giray Pultar, Coulbros Consulting LLC

Infrastructure
What to Do When the Lease Expires: A Moving Experience
Lloyd Cha, Chris Motta, Syed Babar, Mukul Agarwal, Advanced Micro Devices; Jack Ma, Waseem Shaikh, Torex Mountain Software; and Istvan Marko, Volt Services Group
Anatomy of an Athena Station
Karl Ramm and Thomas Bushnell, BSG, MIT Information Systems
Bootstrapping an Infrastructure
Steve Traugott, NASA Ames Research Center and Joel Huddleston, Level 3 Communications

Printing and Configuring Files
Ganymede: An Extensible and Customizable Directory Management Framework
Jonathan Abbey, Applied Research Laboratories of The University of Texas at Austin
Building an Enterprise Print System
Ben Woodard, Cisco Systems, Inc.
Large Scale Print Spool Service
Ignacio Requero, David Foster, and Ivan Delcosse, CERN, European Laboratory for Particle Physics

Invited Talks

Joint Session: Succumbing to the Dark Side of the Force: The Internet as seen from an Adult Website
Dan Klein, Cyberainment, Inc.
The adult industry is by far the biggest consumer of net bandwidth. It is arguably also the largest cash source for content providers. Without getting into the “politics” of the industry as a whole, this talk will examine the many facets of this much-maligned (and hugely subscribed) dark side of the Web. This talk will examine what it means to be in a service industry, advertising, site scaling and bandwidth, monitoring, load sharing, load shedding, and load stealing. It will also look at issues of security, payment methods, billing, theft, risk, and spamming. It will also show how data mining can be a boon and a bane, and look at issues of copyright protection and abrogation. Finally, issues of site automation, what kind of people run adult sites, and how much money can be made will be explored.

Practicum

Branchstart—A Generic, Multi-OS Installation Server
Rory Toma, WebTV Networks Inc.
This talk describes an implementation of a single architecture, multi-OS network installation server. This server has the capability to install to any Intel-based computer, practically any OS that will run on it.

Panel: Technical Summaries

Repetitive Strain Injury (RSI): Causes, Treatment, and Prevention
Jeff Okamoto, Hewlett Packard
This talk will start with the basic anatomy of the body and habits which may cause an RSI to occur. It will then cover the most common types of RSI, including their symptoms, your interactions with your doctor, and various treatment methods. Finally, some tips on dealing with the Worker's Compensation system and ways to prevent RSI will be covered.

Mailer Wars
Which mailer is the best one? Sendmail is the most popular. Omail claims no bugs. Vmail is designed to be a super solution. This session will feature advocates of each of the various mailers defending their favorite server.

Joint Session: The Great Certification Debate

SAGE's initiative to provide certification for system administrators has certainly generated a huge amount of commentary from its members. This debate will ponder the pros and cons of such a plan.

Joint Session: SAGE Community Meeting & Candidates Forum

Find out how to get involved in SAGE, and hear from the SAGE Executive Committee on their recent and upcoming activities. The committee will be on hand to answer your questions, and to solicit your ideas and feedback on how to better serve SAGE members. You will also hear from the candidates running in the upcoming SAGE election. Everybody is welcome.

Visit our web site: http://www.usenix.org/events/lisa98/
12th Systems Administration Conference (LISA '98)
Sunday-Friday, December 6-11, 1998 • Boston, Massachusetts

Friday, December 11

Referred Papers
Distributing Software Packages
Mpkg: A Software Packaging Tool
Carl Staelin, Hewlett-Packard Laboratories
SEPP, Software Installation and Sharing System
Tobias Detiker, Department of Electrical Engineering, Swiss Federal Institute of Technology
SyncTree—For Single-Point Software Installation Upgrades and Patches
John Lockard, University of Michigan; and Jason Larke, ANS Communications

Invited Talks
Overview of the LISA/NT Conference
Ian Reddy, Cisco Systems, Inc.
This session will present the highlights of the Large Installation System Administration of Windows NT Conference, held August 5-8, 1998.

Practicum
Network Admin and Remote Computing
Network administration is a hot topic. This session will discuss network admin. and interfacing off-site workers to a central site. Discussions will include both technical and political issues.

New Thoughts and Evolution
The Evolution of the CMD Computing Environment: A Case Study in Rapid Growth
Lloyd Cha, Chris Motta, Syed Babar, Mukul Agarwal, Advanced Micro Devices; Jack Ma, Waseem Shaikh, Tseng Mountain Software; and Istvan Marko, Volt Services Group Computer Immunology
Mark Burgess, Centre of Science and Technology, Oslo College, Norway
A Visual Approach for Monitoring Logs
Luc Girardin and Dominique Brodie, UBS, Ubiab

Security as Infrastructure
Tom Perrine, San Diego Supercomputer Center
Are you shooting rabbits or building fences? This talk will describe security architectures: reliable, robust, and comprehensive plans to incorporate security into your networks and hosts. Unlike "patch of the day" and other reactive methods, security architectures are implemented to prevent entire classes of security problems throughout your network.

Works-in-Progress (WIPs)
Do you have interesting work you would like to share, or a cool idea that is not yet ready to be published? The LISA audience provides valuable discussion and feedback. We are particularly interested in presentation of student work. To schedule your short report, send email to Peg Schafer, lisa-wips98@usenix.org.

Mailing Lists
Mailman: The GNU Mailing List Manager
John Vega, Reliable Software Technologies; Barry Warsaw, and Ken Manheimer, Corporation for National Research Initiatives
Drinking from the Fire(walls) Hose: Another Approach to Very Large Mailing Lists
Strata Rose, VirtualNet Consulting; Christine Hogan, Greg Kulos, and Bryan McDonald, Global Networking and Computing, Inc.
Request v3: A Modular, Extendable Task Tracking Tool
Joe Rhet, Navigist

Practical Cryptography—Privacy for Business and Electronic Commerce
Frederick M Avolio, Security Consultant
This session will explain cryptographic basics, but concentrate on the tools and methods necessary for privacy for business (or personal) transactions and how they are and will be used in electronic commerce. It is not a technical presentation to discuss technical characteristics of the schemes. Rather, it is a general session to educate the system and MIS managers, who deploy encryption-enabled technology to support business on the Internet.

Palm Pilot Magic
Palm Pilots: Share your business card with someone at a single touch of a button, categorize expense reports in real time, carry your rolodex around with you. This session will discuss not only interesting things to do with your Palm Pilot but also the system administration impacts of trying to synchronize it, e.g., company phone books, while on the road.

Joint Closing Session:
The LISA Quiz Show!

HOSTED BY ROB KOLSTAD

The LISA QUIZ SHOW debuted in Boston many years ago and now makes a triumphant return. Host Rob Kolstad promises new questions and categories. Come to Boston to win prizes and to match your wits with other administrators in the toughest quiz show in the industry.

Fax 1.949.588.9706 for more information
Call for Papers and Participation

2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)

February 25-26, 1999
New Orleans, Louisiana, USA

Sponsored by the IEEE Computer Society's Task Force on Internetworking (TFIW) and Technical Committee on Operating Systems (TCOS)
In cooperation with the USENIX Association

This past decade has seen the widespread adoption of both portable computing devices and wireless communication networks. However, the integration of these two technologies has yet to occur on a large scale. Achieving pervasive mobile computing will require advances in many areas such as:

- Network and operating system support for mobility
- Application adaptation to changing conditions
- Portable computing hardware with networking capability
- Digital audio and video in mobile environments
- Performance evaluation of wireless data networks
- Mobile Internet and Web access
- Security and privacy
- Power management

Following the example of the first WMCSA (see http://www.cs.odu.edu/~mukka/tcos/arch/spring95/spring95.html), the goal of this workshop is to foster the exchange of ideas in mobile computing among workers in the field. Attendance will be limited to about 60 participants, based on the position papers submitted. We seek papers that describe ongoing or completed research and development efforts. We are particularly interested in papers that propose new directions, advocate non-traditional approaches, or generate controversy and discussion. Submissions must not exceed 10 pages in length. We will publish a printed proceedings.

A small number of graduate students will be granted a waiver of the registration fee. In return, these students will be asked to take notes at the workshop. Students who wish to be considered for the waiver must send in a brief description of their current research, and an explanation of how participation in the workshop is likely to help them.

WMCSA '99 will take place immediately following and in the same location as the 3rd USENIX Symposium on Operating Systems Design and Implementation (OSDI '99). Please see http://www.usenix.org/events/osdi99 for more information on OSDI '99.

ORGANIZERS
General Chair Sumi Helal, U. of Florida
Finance Chair Mukesh Singhal, Ohio State U.
Publications Chair Joe Duran, Motorola
Publicity Chair Jin Jing, GTE Labs
Local Arrangements Chair Golden G. Richard, III, U. of New Orleans

Program Chair Ramon Caceres, AT&T Labs
Program Committee Mary Baker, Stanford U.
B. Badrinath, Rutgers U.
Nigel Davies, U. of Lancaster
Dave Johnson, Carnegie Mellon U.
Anthony Joseph, U.C. Berkeley
Jay Kistler, FORE Systems
Karin Petersen, Xerox PARC
Steve Pink, Luella U.
Sriniv Seshan, IBM Research
Cormac Sreenan, AT&T Labs

Steering Committee
Ramon Caceres, AT&T Labs
Fred Douglass (Chair), AT&T Labs
Sumi Helal, U. of Florida
David Kotz, Dartmouth College
Darrell Long, U.C. Santa Cruz

SUBMISSION INSTRUCTIONS
Send a PostScript or PDF copy of your position papers, at most 10 pages in length, to wmcsa99@research.att.com.
Applications for student waivers should be sent to the same address.

IMPORTANT DATES
Submissions due October 16, 1998
Acceptance notification November 23, 1998
Camera-ready copy due December 18, 1998

ADDITIONAL INFORMATION
For the latest information on WMCSA'99, please see http://www.research.att.com/conf/wmcsa99
April 11-12, 1999
Santa Clara Marriott Hotel
Santa Clara, California

Important Due Dates for Refereed Paper Submissions
Extended abstracts due: November 1, 1998
Notification to authors: November 23, 1998
Full papers for editorial review: December 12, 1998
Camera-ready full papers: February 23, 1999

Program Committee
Chair: Marcus J. Ranum, Network Flight Recorder
Charles Antonelli, University of Michigan
Frederick Avolio, Avolio Consulting
Tina Darnhofer, System Experts
Dan Geer, CERTCO
Norm Laudermilch, UUNet/Worldcom

Overview
The goal of this workshop is to bring together network managers, engineers and researchers interested in deploying and developing intrusion detection systems (IDS) and network monitoring technologies for security, traffic analysis, or forensics. The workshop will emphasize practical results, case studies, and real-world large-scale deployment of ID.

This will be a two-day workshop, consisting of invited talks, refereed papers, and work-in-progress reports. Opportunities to get together informally will include a Sunday evening hosted reception, a hosted lunch on Monday with Marcus Ranum chairing, and Birds-of-a-Feather sessions on Sunday.

Technical Sessions
Intrusion detection offers the promise of automatic detection and notification of break-ins or unauthorized use of computers. With networks becoming increasingly interconnected, it's difficult to draw a clear boundary between "internal" and "external" — better techniques for detecting abuse from within are becoming mandatory.

We seek papers describing original work concerning the design, implementation, and real-world application of intrusion detection and network monitoring technologies. Besides mature work, we encourage submissions describing exceptionally promising prototypes, or enlightening negative results. Case studies and experience papers are particularly of interest. Share your results, share your pain, share your ideas.

Where appropriate, authors will be able to demonstrate their applications during their presentation using systems that will be fed with packets captured at "live" sites, which contain various intrusion attempts. Also, space will be available to authors to demonstrate their work outside of their presentation in a more relaxed and interactive environment.

Topics
Topics of interest include, but are not limited to:

- Case studies of IDS in practice
- Statistical models for IDS
- Anomaly detection systems
- Misuse detection systems
- Host based approaches to IDS
- Network based approaches to IDS
- Application-based approaches to IDS
- IDS in cryptographically protected networks
- Distributed IDS in large networks
- Correlation techniques
- Event thresholding
- Reducing false positives
- Alternative approaches

What To Submit
Authors must submit an extended abstract by Nov 1, 1998. This should be 5-7 pages long or about 2500-3500 words, not counting references and figures. You may submit a full paper for use by the program committee if there are questions about the abstract, but the full paper is not required. Longer submissions, not accompanied by an appropriate extended abstract, will be penalized in the review process.

The full papers resulting from accepted abstracts will go through an editorial review cycle with a member of the program committee, and should end up about 10-12 pages long.

The objective of an extended abstract is to convince the reviewers that a good paper and 25-minute presentation will result. It is important to identify what has been accomplished,
to explain why it is significant, and to compare with prior work in the field, demonstrating knowledge of the relevant literature. The extended abstract should represent the paper in “short form.” It must include the abstract as it will appear in the final paper. The body of the extended abstract should be complete paragraphs, not just an outline of the paper. (Sections present in the full paper but omitted from the abstract may be summarized in terse form.) Authors should include full references, figures when available, and as is usually appropriate, performance data. Such data also help indicate the status of the implementation, often a crucial issue. The abstract will be judged on significance, originality, clarity, relevance, and correctness.

This Workshop, like most conferences and journals, requires that papers not be submitted simultaneously to another conference or publication and that submitted papers not be previously or subsequently published elsewhere. All submissions will be held in the strictest confidence prior to publication. Papers accompanied by so-called “non-disclosure agreement” forms are not acceptable and will be returned unread.

Please read the detailed author guidelines. For a copy of the author guidelines, either see http://www.usenix.org/events/detection99/guidelines.html, or send email to detection99authors@usenix.org.

How To Submit

The current Call for Papers is preliminary. Deadlines and instructions may undergo minor changes. Authors should check the USENIX Association Web page (www.usenix.org) for final instructions after October 15.

Submissions should be done electronically via email. Make submissions to detection99papers@usenix.org.

A form will be provided on the Web to facilitate submission and ensure that all submissions provide all the information that is required. This information will include:

1. The title of the paper and the names and affiliations of all authors. (Note: authors’ names and affiliations will be known to the reviewers).
2. The name, email and postal addresses, day and evening phone numbers, and a fax number if available, of one author who will serve as a contact.
3. An indication of which, if any, of the authors are full-time students.

The alternate method of submission will remain postal mail; you may mail 15 copies of your submission to:

Marcus J. Ranum
16400 Ed Warfield Rd
Woodbine, MD 21797
410-489-4995

All submissions will be acknowledged electronically; you must provide an email address. If you have not received an acknowledgment within 60 hours of submitting your abstract electronically (or the submission deadline, if the submission is early), please contact the program chair at detection99chair@usenix.org.

Work-In-Progress Reports

Do you have interesting work you would like to share, or a cool idea that is not ready to be published? Works-in-progress reports are for you! Works-in-progress reports, scheduled during the technical sessions, introduce new or ongoing work. The USENIX audience provides valuable discussion and feedback. We are particularly interested in presentations of student work. To schedule your report, please contact the Works-in-Progress coordinator at detection99wips@usenix.org.

Program/Registration Materials

Materials containing all details of the technical program, registration fees and forms, and hotel information will be available online at http://www.usenix.org/events/detection99/ and in print in January 1999. If you wish to receive the printed program materials, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630 USA
Phone: +1 949-588-8649
Fax: +1 949-588-9706
Email: conference@usenix.org

Rev. 8-18-98
Preliminary Call for Papers

2nd USENIX Symposium on Internet Technologies and Systems
Sponsored by USENIX, the Advanced Computing Systems Association
Co-sponsored by the IEEE Computer Society Task Force on Internetworking (pending)

October 11-14, 1999
Regal Harvest House Hotel
Boulder, Colorado, USA

Note: Deadlines and instructions may undergo minor changes. Authors should check for final instructions after October 15, 1998.

Important Due Dates for Refereed Paper Submissions
Extended abstracts due: April 15, 1999
Notification to authors: May 28, 1999
Full papers for editorial review: July 23, 1999
Camera-ready full papers: August 31, 1999

Symposium Organizers
Program Chair
Fred Douglass, AT&T Labs—Research
Program Committee
Eric Brewer, University of California Berkeley and Inktomi
Dan Connolly, W3C
Peter Honeyman, University of Michigan
David B. Johnson, Carnegie Mellon University
P. Krishnan, Bell Labs, Lucent Technologies
Geoff Kuenning, University of California Los Angeles
Yoelle Maarek, IBM Haifa Research Lab
Udi Manber, University of Arizona
Jeffrey Mogul, Digital Equipment Corp. Western Research Lab
Katia Obraztza, Information Sciences Institute

Overview
The goal of this symposium is to bring together engineers and researchers interested in developing innovative Internet applications and technology.

This will be a 3.5 day symposium, with 1 day of tutorials, followed by 2.5 days of refereed paper presentations, invited talks, works-in-progress presentations, demos, panel discussions, and Birds-of-a-Feather sessions.

Tutorials, October 11, 1999
Tutorials for technical staff, researchers, managers, and students will provide immediately useful, practical information on topics such as Web security, XML, and Internet performance.

If you are interested in proposing a tutorial, contact the USENIX tutorial coordinator, Dan Klein, by phone at +1.412.422.0285 or by email to dsk@usenix.org

Technical Sessions, October 12-14, 1999
The Internet continues to evolve in interesting and unexpected ways. Electronic commerce, mobility, streaming media, and other developments are driving the creation of new applications, protocols, security models, and systems. Recent application-level changes, such as XML, may dramatically improve the functionality of the Web. What's next?

USITS emphasizes both innovative research and quantified experience in Internet applications, technologies, and systems. We seek papers describing original work concerning the design, implementation, and application of Internet technologies.

Besides mature work, we encourage submissions describing exceptionally promising prototypes, or enlightening negative results. Case studies and experience papers are particularly of interest.

Where appropriate, authors will be able to demonstrate their applications during their presentation using computers linked to the audio-visual system and the Internet. Also, space will be available to authors to demonstrate their work outside of their presentation in a more relaxed and interactive environment.

Topics
Topics of interest include, but are not limited to:

Distributed caching and replication
Electronic Commerce
IPv6
Information indexing/retrieval/management
Internet agents
Java, Inferno, Safe-Tcl, Python, and other "Internet programming" tools
Performance (network, server, end-to-end)
Resource discovery
Security

Note that just because a paper is about Java (for instance) does not mean it pertains to the Internet. Off-topic papers will be referred to other forums.
Best Paper Awards
Awards will be given for the best paper and best student paper at the conference.

What To Submit
Authors must submit an extended abstract by April 15, 1999. This should be 5-7 pages long or about 2500-3500 words, not counting references and figures. In addition, you may submit a full paper for use by the program committee if there are questions about the abstract, but the full paper is not required. Longer submissions, not accompanied by an appropriate extended abstract, will be penalized in the review process. The full papers resulting from accepted abstracts will go through an editorial review cycle with a member of the program committee, and should end up about 10-12 pages long. Very similar papers must not have been published or concurrently submitted for publication elsewhere.

The objective of an extended abstract is to convince the reviewers that a good paper and 25-minute presentation will result. It is important to identify what has been accomplished, to explain why it is significant, and to compare with prior work in the field, demonstrating knowledge of the relevant literature. The extended abstract should represent the paper in "short form." It must include the abstract as it will appear in the final paper. The body of the extended abstract should be complete paragraphs, not just an outline of the paper. (Sections present in the full paper but omitted from the abstract may be summarized in terse form.) Authors should include full references, figures when available, and as is usually appropriate, performance data. Such data also help indicate the status of the implementation, often a crucial issue. The abstract will be judged on significance, originality, clarity, relevance, and correctness. All submissions will be held in the highest confidence prior to publication. Papers accompanied by so-called "non-disclosure agreement" forms are not acceptable and will be returned unread.

If you would like to receive detailed guidelines for submission and examples of extended abstracts, send email to: usits99authors@usenix.org or telephone the USENIX Association office at +1.510.528.8649.

How To Submit
The current Call for Papers is preliminary. Deadlines and instructions may undergo minor changes. Authors should check for final instructions after October 15 at the URL <http://www.usenix.org/events/usits99>.

Submissions should be done electronically. A form will be provided on the Web to facilitate submission and ensure that all submissions provide all the information that is required. This information will include:
1. The title of the paper and the names and affiliations of all authors. (Note: authors' names and affiliations will be known to the reviewers).
2. The name, email and postal addresses, day and evening phone numbers, and a fax number if available, of one author who will serve as a contact.
3. An indication of which, if any, of the authors are full-time students.

The alternate method of submission will remain postal mail; you may mail 15 copies of your submission to:

Fred Douglish
Room B137
AT&T Labs-Research
180 Park Avenue
Florham, NJ 07932-0971
phone: +1.973.360.8775

All submissions will be acknowledged electronically; you must provide an email address. If you have not received an acknowledgment within 48 hours of submitting your abstract electronically (or the submission deadline, if the submission is early), please contact the program chair at douglis@usenix.org.

Work-In-Progress Reports
Do you have interesting work you would like to share, or a cool idea that is not ready to be published? Works-in-progress reports are for you! Works-in-progress reports, scheduled during the technical sessions, introduce new or ongoing work. The USENIX audience provides valuable discussion and feedback. We are particularly interested in presentations of student work. To schedule your report, please contact the Works-in-Progress Coordinator at usits99wips@usenix.org.

Registration Materials
Materials containing all details of the technical and tutorial programs, registration fees and forms, and hotel information will be available online at http://www.usenix.org/events/usits99/ and in print in August 1999. If you wish to receive the printed registration materials, please contact USENIX at:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630 USA
+1 949-588-8649;
Fax +1 949-588-9706
email: conference@usenix.org
Conferences

I am getting pretty excited about this upcoming LISA conference (December 6-11 in Boston). I volunteered to co-chair it with Xev Gittler and it looks like it's turning out to be a humdinger.

The USENIX staff is doing its usual fabulous job of publicizing everything, organizing rooms and hotels, making sure the conference runs as smoothly as do so many USENIX events.

Xev and the program committee have assembled a fabulous set of papers, with lots of new things this year. The invited talks look to be something great again – they always attract a huge audience.

The Advanced Topics Workshop is held on Tuesday again this year: new developments, issues, technical, political, the entire garnet. It's a limited-attendance gig for those who wish to write a short position paper.

The vendor exhibits promise to be huge with all the latest commercial products on display.

And this year, there are some new things.

First of all, there's an entire new track. This track will include all sorts of pragmatic and interesting talks from presenters who are both knowledgeable and able to present their material in an interesting way. This track turns out to increase the available technical presentations for the attendees by 50%.

On Monday, there's another limited-attendance workshop. The first Global-LISA Workshop is for those concerned with the issues of running huge sites' networks in globally distributed ways. An experimental, highly-interactive, limited-attendance workshop, its goal is to bring together those with common interests in solving problems related to highly distributed corporate environments. This event also seeks a position paper.

Of course, the evenings include the usual range of BOFs, entertainment, social gatherings, and general fun goings-on.

We'll close this year with the LISA Quiz Show. I am not sure if Hal Pomeranz will be there to defend his title, but if he is: look out because he is a pro at the Quiz Show.

Why do I bring this up? Because it's really a great thing to go to a conference once in a while.

First of all, it gives one a chance to look at a bigger version of the picture than one gets in the day-to-day computer world. By joining a remote gathering of peers, you can put your own good work in perspective and hear what other people are doing. It's refreshing; it's different; it's valuable. Of course, unlike what some people think, it's also fairly hard work.

Second, you get to see the problems that are perceived to be on the horizon (often with solutions). You get to compare your solutions to those of other admins. You get to rub shoulders with the entire spectrum of sysadmins.

Third, you get to bounce your ideas off the other attendees, attend BOFs for your particular interests, and have the best technical dinner chat in these fine United States. The combination of the synergy, knowledge, and format (including tutorials) combines to make a week whose value is very high in both gained knowledge, increased perspective, and recharged batteries.

I love conferences. I hope you'll consider coming to LISA – it's going to be a blast.
Have All The Tools You Need

Order the Performance Computing UNIX System Administration Toolkit CD-ROM

This low-cost comprehensive CD-ROM is packed with tools for the UNIX system administrator. This updated edition includes the latest versions of these tools, thus saving you hours of search and download time. It also includes source code, select software demos from leading solutions providers, and a library of reference material from past issues of UNIX Review and Performance Computing.

Check out these features*

Programming Tools and Languages
Data Communication Tools
Security Tools
Performance Monitoring Tools
Web Server Software
Web Browsers

*For a complete list of included applications go to www.performance-computing.com

Plus...

Select Solutions-oriented Product Demos and a Comprehensive Library of UNIX Review and Performance Computing Editorial Content Including an Archive of System Management Feature Articles “Perl Advisor” and “Daemons & Dragons” Columns

ALL FOR JUST $39.95!

ORDER FORM

YES! Rush me ____ copies of the UNIX System Administration Toolkit CD-ROM at $39.95 each!

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$39.95</td>
<td></td>
</tr>
</tbody>
</table>

Subtotal

Shipping/Handling

Sales Tax

TOTAL

Complete this form and return to: UNIX System Administration Toolkit CD-ROM, 1601 West 23rd Street, Suite 200, Lawrence, KS 66046. Telephone: 800/444-4881 or 913/841-1631; FAX 913/841-2624; e-mail orders@mfi.com

Name

Company Name

Address

City State Zip Country

Phone Fax

☐ Check ☐ VISA ☐ MasterCard ☐ American Express

Credit Card# Expiration Date

Signature

Sales Tax: Residents of NY-8%, IL-6.25%, CA-7.5%, TX-7.5%, KS-6.9%, GA-6%
Shipping & Handling: $4 for U.S. and Canada orders; $10 for international orders
MEMBERSHIP AND PUBLICATIONS
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITE
http://www.usenix.org

PGP INFORMATION
All correspondence to:
1998 Operational Key
Key ID: 1024/F6F82613 1997/11/21
USENIX 1998 <pgp@usenix.org>
Key fingerprint = 80 6F B5 48 C2 1A B8 45
48 5F 2F 38 E6 41 B0 61

1998 Signing Key
Key ID: 1024/4F75A901 1997/11/21
USENIX 1998 Signature
<http://www.usenix.org/pgp/pgpsig.html>
Key fingerprint = 05 FD CF A1 2F 47 00 5C
69 C2 25 E4 66 89 A6 98

USENIX master key <not-for-email>:
Key ID: 1024/2FEA2EFI 1996/04/08
Key fingerprint = DB A7 50 99 66 E4 8A A9
80 B2 D9 E2 FE DA 00 5E

CONTRIBUTIONS SOLICITED
You are encouraged to contribute articles, book reviews, and announcements to ;login:. Send them via email to <login@usenix.org> or through the postal system to the Association office.

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of ;login: are December 1, 1998, and February 2, 1999.

ADVERTISING ACCEPTED
;login: offers an exceptional opportunity to reach 9,000 leading technical professionals worldwide.

Contact: Cynthia Deno
cynthia@usenix.org
408 335 9445