## Upcoming Events

**6th International Python Conference**  
Sponsored by CNRI and the PSA and co-sponsored by USENIX  
**WHEN** | **WHERE** | **WHO program co-chairs**  
--- | --- | ---  
October 14-17/97 | San Jose, CA | Guido van Rossum  
                |                | CNRI  
                |                | Jeremy Hylton  
                |                | CNRI

**Conference on Domain-Specific Languages**  
**WHEN** | **WHERE** | **WHO program chair**  
--- | --- | ---  
October 15-17/97 | Santa Barbara, CA | Chris Ramming  
                    |                | AT&T Research

**11th Systems Administration Conference (LISA '97)**  
Co-sponsored by USENIX and SAGE  
**WHEN** | **WHERE** | **WHO program co-chairs**  
--- | --- | ---  
October 26-31/97 | San Diego, CA | Hai Pomeranz  
                |                | Neimark  
                |                | Celeste Stokely  
                |                | Stokely Consulting

**USENIX Symposium on Internet Technologies and Systems**  
**WHEN** | **WHERE** | **WHO program chair**  
--- | --- | ---  
December 8-11/97 | Monterey, CA | Carl Staelin  
                    |                | Hewlett-Packard Laboratories

**DEADLINES**  
Final Papers  
October 22/97

**7th USENIX Security Symposium**  
**WHEN** | **WHERE** | **WHO**  
--- | --- | ---  
January 26-29/98 | San Antonio, TX | Avi Rubin, Program Chair  
                  |                | AT&T Labs - Research  
                  |                | Greg Rose, Invited Talks Coord.  
                  |                | Qualcomm Australia

**DEADLINES**  
Final Papers  
December 9/97

**New Network Technologies Symposium**  
**WHEN** | **WHERE** | **WHO program chair**  
--- | --- | ---  
March 2-3/98 | Salt Lake City, UT | Phil Winterbottom  
                  |                | Bell Labs

**DEADLINES**  
Notification to Authors  
Full Papers  
Final Papers  
November 3/97 | December 3/97 | January 19/98

**4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)**  
**WHEN** | **WHERE** | **WHO program chair**  
--- | --- | ---  
April 27-30/98 | Santa Fe, NM | Joe Sventek  
                  |                | Hewlett-Packard

**DEADLINES**  
Tutorial Submissions  
Paper Submissions  
Notification to Authors  
Final Papers  
December 2/97 | December 2/97 | January 7/98 | March 17/98

**USENIX Annual Technical Conference**  
**WHEN** | **WHERE** | **WHO**  
--- | --- | ---  
June 15-19/98 | New Orleans, LA | Fred Douglass, Program Chair  
                  |                | AT&T Labs  
                  |                | Invited Talks Coordinators:  
                  |                | Clem Cole  
                  |                | Digital Equipment Corporation  
                  |                | Barry Kercheval  
                  |                | Xerox PARC

**DEADLINES**  
Paper Submissions  
Notification to Authors  
Full Papers  
Final Papers  
November 25/97 | January 26/98 | March 30/98 | April 27/98

**6th Annual Tcl/Tk Conference**  
**WHEN** | **WHERE** | **WHO program co-chairs**  
--- | --- | ---  
September 14-18/98 | San Diego, CA | Don Libes  
                  |                | NIST  
                  |                | Michael J. McLennan  
                  |                | Bell Labs

**3rd USENIX Workshop on Electronic Commerce**  
**WHEN** | **WHERE** | **WHO program chair**  
--- | --- | ---  
August 31-Sept. 3/98 | Boston, MA | Mark Manasse  
                  |                | DIGITAL Systems  
                  |                | Research Center

**2nd USENIX Windows NT Symposium**  
**WHEN** | **WHERE** | **WHO program chair**  
--- | --- | ---  
Summer/98 | Seattle, WA | Thorsten von Eicken  
                  |                | Cornell University

**2nd Large Installation System Administration of Windows NT Conference**  
**WHEN** | **WHERE** | **WHO program co-chairs**  
--- | --- | ---  
Summer/98 | Seattle, WA | Remy Evard  
                  |                | Argonne National Laboratory  
                  |                | Ian Reddy  
                  |                | Simon Fraser University
IN THIS ISSUE . . .

LETTERS TO THE EDITOR
3 Another Disk Usage Tool from Thomas Richter

USENIX NEWS
5 Letter from the President
5 More Letters About "Whither USENIX?"
7 USA Wins Gold at Central European Programming Contest by Rob Kolstad
8 Twenty Years Ago in UNIX NEWS by Peter H. Salus
9 Deadline for Student Proposals
9 1998 USENIX Nominating Committee by Ellie Young
10 Call for Tutorial Proposals by Dan Klein

FEATURES
30 Musings by Rik Farrow
33 Discussing Security: An Interview with Charlie Modell and Mark Teicher by Rob Kolstad
40 The Web Master: Surveys Online! by Dave Taylor
45 Using Java by Terry Slattery
49 MrMean the Hacker by Peter V. Radatti
52 Using C++ as a Better C by Glen McCluskey

CONFERENCE REPORTS
55 Are You Tcl-ish? by Peter H. Salus
56 Reports on the Fifth Tcl/Tk Workshop by Brian Bailey

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

BOOK REVIEWS
70 The Bookworm by Peter H. Salus
72 Web Security Sourcebook Web Security and Commerce Reviewed by Rik Farrow
73 Java Security: Hostile Applets, Holes, and Antidotes Reviewed by George W. Leach
74 Web Client Programming with Perl Reviewed by Carolyn Sienkiewicz
75 Java AWT Reference Reviewed by Bruce O’Neal
76 Internet Economics Reviewed by Terry Rocker
77 sendmail, 2d ed. Reviewed by Steve Hanson

ANNOUNCEMENTS & CALLS
78 LISA XI
80 Conference on Domain-Specific Languages

LOCAL USER GROUPS
86 motd
by Rob Kolstad
Welcome to a new, improved look for ;login:. For the past five months the staff and Vinje Design have been working on a makeover that we believe will make each issue more readable as well as easier to navigate. In future issues we will be adding more photos, new features and special issues. We are also planning to put more material on our Web site after it has appeared in print. (And if you haven’t looked at the site lately, it also has been redesigned to make it easier and faster to get around, plus you can now register online to attend our conferences.)

Inside you’ll find more feedback from the membership on “Whither Now USENIX?” Our historian/chronicler of UNIX and USENIX, Peter Salus, offers the first installment of a regular column, “20 Years Ago...”. A new Java columnist explains how he built the Java version of TTCP. An article from the founder of Cybersoft gives advice on how to deal with “social engineering” attacks and offers policies on how to deal with this problem.

Tina Darmohray, the editor for SAGE, has pulled together a greatly expanded Features section that begins with an article by Bruce Markey of Netscape on LDAP. It is followed by the latest tips from ToolMan, a reliability article on network and network service planning, an overview of ActiveX, and advice/URLs to help alleviate the woes of Spam.

By providing summaries of the USENIX conferences, we hope you will get an overview of what transpired during the sessions (although these don’t capture the action in the halls, BOFs, and taverns). Check out the very thorough, excellent reports from the Tcl/Tk Workshop by Brian Bailey and Peter Salus.

There is also an important update from our standards representative, Nick Stoughton, about big changes in the POSIX standards arena, as well as the changes in direction/focus of the Open Group.

And last, our reviewers have been busy this past summer reading, and this is one of the biggest Bookworm/review sections ever!

We would like to hear from you what you think about the design and content (and of course if you wish to contribute that’s even better!).

PS: We’re working on a special special issue on NT, edited by Rik Farrow, to arrive in your mailbox soon!
Another Disk Usage Tool

I just read with interest the article “Toolman Visualizes Disk Usage” by Daniel E. Singer in the ;login: August 1997 issue. He addressed the problem of tracking down who uses (or sometimes wastes) how much disk space. He also presented a tool to help users and system administrators to identify people consuming a lot of disk space.

Being a system administrator myself for nearly ten years now, I was faced with similar problems before and have developed a simple tool achieving a similar result. However, the approach is different. I find this tool very useful and would like to present it to you. While the tool is similar to the quota system reporting tools, it requires neither their disk space nor overhead.

The tool (named diskusage) is a small korn shell script invoked by cron every night. It runs with root privilege. The principle is very simple: diskusage gets a list of mounted file systems and records the number of blocks in use for each file system. File systems mounted from remote hosts via AFS or NFS are ignored, since those are someone else’s problem.

For each local file system, the tool records file ownership and accumulates the file sizes per owner. At the end of the file system traversal a list is printed on a file system basis, naming each user and the amount of disk space used on a file system. The tool prints the amount of disk space needed in relation to the total number of file system blocks in use (not the file system size).

```
#!/usr/bin/ksh
# Thomas Richter, IBM Boeblingen, Germany, Feb-95
# Program for disk space usage. Read the local
# filesystem names and verify how much disk space
# is used by each user.
#
PATH=/usr/bin

# Get local file systems (mount points) without
# NFS and AFS remote system
fslist=$(df -i)

# Get all files on local filesystem and store
# (one file per line) in tmp file
# filesystem filename size(KB) userid permissions
# (one file per line) in tmp file
# filesystem filename size(KB) userid permissions
for fs in $fslist
do
    find $fs -xdev -fstype jfs -ls 2>/dev/null | awk '{ print $7 }'
    $3 = ~/[-dls]/ { printf("%s %s %d %s\n",
    fsname, $1, $2, $5, $3) }
    $3 = ~/^[chbp]/ { printf("%s %s %d %s\n",
    fsname, $12, $12, $5, $3) }
done

# Get amount of storage used per user and print
# statistics.
for fs in $fslist
do
    fssize=$(df -i $fs | awk '{ print $3 }')
grep "$fs " $tmpfile | awk '{print fssize="$fssize"}'
    # For each record get owner, and size in bytes.
```

To reduce the output to a practical length, diskusage doesn’t report users occupying less than 1% of the file system size in use.

The tool also does some basic security checks, such as looking for:
- orphaned files (files whose owner or group they belonged to have been removed from the system)
- device special files not in the /dev directory

The script mails the outcome to the system administrator. It gives a good overview about your users’ habits.

Since the tool works on a file system basis, it also helps prevent problems with file systems users are not immediately aware of such as /var/spool (for mail reception and spooling) and /tmp. It helps identify people not reading their mail (for whatever reason) or who have lots of files in /var/spool/preserve and don’t know about it.

The korn shell script follows this letter. It also uses awk. There is also a perl script available from the author. I hope you enjoy reading it, and I look forward to hearing from you soon.

Thomas Richter
IBM Boeblingen, Germany
<richter@vnet.ibm.com>
more letters...

```bash
{ size[$4] += $3 }
END {
    for ( u in size ){
        percent = size[u]/fssize*100
        if( percent < 1 )
            continue;
        printf "%-25s %5.2f%% % 5dmb\t%s\n", fsname, percent, size[u]/1024, u
    }
}
done | sort -k1,1d -k2r

# Special files not in /dev
awk '$5 != /^bc$/ && $2 != /^\// { print $2 " \ $5 }' $tmpfile >$(tmpfile)1
print "Device files NOT in /dev: $(cat $(tmpfile)1 | wc -l)"

# Searching for setuid/setgid files owned by root
# or group system
# The list of files must then be checked against
# all entries in the syschk database. Files not
# listed there must be reported!!!!
# Run the syschk program to verify important
# files and devices

# Remove temporary files
rm -f $tmpfile $(tmpfile)1

# DO other checks:
# Searching for orphaned files
find $falist -xdev -nouser -ls >$(tmpfile)1
print "\nOrphaned files$(cat $(tmpfile)1
    | wc -l) owned by"
awk '{ print $5 }' $(tmpfile)1 | sort | uniq
```

DOCTOR FUN

![Doctor Fun Cartoon](image)

Copyright © 1997 David Farley
I'd like to share some of my own feelings on the issues raised by the many responses, published in this and the August issue, to my column, "Whither Now, USENIX?"

I think USENIX is about understanding stuff, making stuff work, and coercing stuff to do new things. USENIX has always been pragmatic about what "stuff" is. We've covered all kinds of operating systems, file systems, programming languages, machine architectures, programming tools, and applications. I've been at USENIX (and AUUG and EUUG) talking where the speakers have passed around stuff to look at, including microprocessors, core memory, geometric models and parts of an artificial kidney.

Some of the letters we received have condemned USENIX for dancing with the devil. Currently, this means Microsoft, but there have been others in the past. Right now, I am doing battle with MVS. And every time I bloody my head running into yet another roadblock set up by MVS's arcane system of file and record types, I mutter "so that's why Ken wrote his file system." Every time I go off and work on other operating systems, I always come back to UNIX. Sure, the flavor changes: System 5, BSD, Plan 9, Linux, UWIN running on Windows NT, but it's still UNIX. And although some folks feel it's a little unfashionable, I am proud of the fact UNIX is the best environment for getting work done that I have used.

I am proud of Ken and Dennis and their Bell Labs colleagues, and the various parts of the extended community including Berkeley, the overseas user groups, and the commercial folks) that have worked on UNIX since 1968.

I work on UNIX every day because I want to, not because I have to; and this has been true since 1976. USENIX will always cover a wide range of computing, because that is what our members want, and need. But UNIX is a core piece of what USENIX covers. I am proud of that, and I think USENIX should be too.

More Letters About "Whither Now, USENIX?"

Andrew,

My apologies for not responding earlier to your "Whither Now, USENIX?" article (6/97).

My credentials and disclaimers: I've worked with computers since 1980 and UNIX since 1985. I view Microsoft as the "Evil Empire," Mr. Gates as the "Anti-Christ," and most of who are rushing to WinNT as lemmings. However, what I think about WinNT is irrelevant, because the market has embraced it so strongly. At home, I run RedHat Linux and Win95 (mea maxima culpa).

My short points.

1. I think it's great that USENIX is thinking about its own future and not resting on its laurels.
2. Bringing the Linux crowd into USENIX in one form or another is a great idea.
3. Focusing on the sysadmin issues of integrating & interfacing with non-UNIX PCs is an unavoidable and a necessary evil.
4. I hope USENIX never loses its focus on UNIX.

Now, I'm free to blather a bit.

There is a need for an organization like USENIX for UNIX. More so, because a lot of other UNIX sources have dried up over the past few years. If USENIX drifts away from UNIX, then I believe another org will come about to replace it. Saying this I can't see USENIX not continuing to support and promote UNIX.

UNIX and USENIX are a lot of different things to a lot of different people. Over time I believe many of our own perspectives of both have changed or at least have been refined. UNIX continues to mature. But it also continues to change and grow. The ability to change and adapt has always been part of UNIX. This has been one its great strengths. As the number of UNIX users continues to grow, they bring with them their own experiences. This too feeds the change process in UNIX. (Just because someone is new to UNIX, doesn't necessarily mean they are new to computing.) The UNIX of today is different than that of the early 90s and the 80s. Likewise the UNIX in the next century will have its differences too. So for those who bemoan the changes to UNIX, this is just part of the game.

With regards to incorporating the Linux User into USENIX, as I said above this is a great idea. They bring with them a lot of fresh enthusiasm and ideas. As group I was very impressed with them at the last USENIX (Anaheim '97). I enjoyed the Linux session I attended. I can't see USELINUX being anything but a plus to USENIX. (Anyone who has used Linux realizes it is more UNIX, i.e., GNU stuff than just Linux. So the differences are not that big!) Besides, given a choice for world domination, I'd prefer Linux win, vice Microsoft!

One last comment on the Linux crowd. One of your respondents felt the UNIX old-timers should embrace the Linux crowd in order to guide them past previous mistakes. I disagree. I say give them the opportunity to fail. Because in doing so, they may find ways to make Linux succeed, where UNIX hasn't.
WinNT is here today, because UNIX (a.k.a. UNIX OEMs) couldn’t or wouldn’t “really” get together to kill it off years ago. Having to integrate WinNT is a given. So in response to those criticizing UNIX Orgs/Mags for taking on the WinNT integration issues, I have to ask, where do you want to learn your WinNT integration from? Microsoft? Or from your UNIX peers? I’d much rather commiserate over the evils of WinNT with my UNIX friends. Learning how to live with WinNT doesn’t mean you’ll have to like it.

Henry E. Alubowicz
<aahu@slip.net>

Andrew Hume’s Response:
I’m glad you made the effort to reply. I agree with many of your points; I’ll respond to a couple of them. UNIX, at least in the form of the POSIX standards, will be a dominant environment for leading edge and research computing systems for several years to come and so it will remain one of USENIX’s major focuses.

I personally couldn’t agree more about letting folks make their own mistakes. I think the point was not that we couldn’t let Linux (or whoever) make their own mistakes, but rather that they do so with full knowledge of the problems with, and consequences of, the various known solutions; the issue is communication rather than control.

Andrew Hume
andrew@usenix.org

Re: The Question
I just got my August of :login: today; so, obviously, I am already too late to answer the original question. But I had a funny feeling when I first read the question, because

- I read it in the first issue of :login: I got because
- I just joined USENIX recently, but at the same time
- I am a Linux user and sysadmin (okay, I did admin AIX and SCO boxes, but those were side jobs, quite some time ago), and
- I have to admin some NT boxes (more in number than Linux boxes, and the number is increasing).

(Okay, my Linux and NT admin duties are still side jobs, at least on paper, but I am having more responsibility than when I administered AIX and SCO boxes.)

And I joined USENIX because I thought that joining SAGE is a Good Thing for a Linux sysadmin.

I don’t know how many people joined USENIX because of the same reason as mine, but perhaps a lot of Linux/FreeBSD users already joined USENIX without anyone noticing. After all, I wasn’t required to disclose what kind of UNIX system I am administering when I joined.

Granted, I wouldn’t have joined USENIX if I only used it at home; but there are businesses that use Linux for essential services.

Ambrose Li
<acli@aci.interlog.com>

<rant_alert>
NO to NT content. I am in favor of open systems. Apart from the interest value of knowing how the system works and contributing to its development, I have professional reasons. My skills stay current across many OSs, allowing me to get a deeper understanding and saving me much leisure time.

NT is a very closed system for Microsoft’s commercial reasons. A customer of Microsoft might have valid financial reasons for using NT, but must factor in the cost of re-educating the sysadmins, programmers, users, support, tech writers.... If as a sysadmin or programmer, you are forced to use NT, then it is fair to expect that your employer pick up the tab for training courses, reference materials, and support. This alters the economic
demand curve for Microsoft products. When USENIX, a non-profit and cooperative organization, organizes NT conferences, or recognizes NT in any manner, it is giving free support to Microsoft. To the extent that the NT market grows as a result of this, USENIX is doing a disservice to its open systems practitioners.

Rick Leir<br/cgi@enterprise.on.ca>

USA Wins Gold at Central European Programming Contest

by Rob Kolstad

Rob Kolstad, editor of *login*, and president of BSDi, is head coach of the USA Computing Olympiad Team.

The USA computing team scored a stunning victory in the Central European Computing Olympiad (CEOI) held July 21-26, 1997, in Nowy Sacz, Poland. The CEOI is widely known for its difficult problems and tough competition, with favorite Romania winning many contests. Invited for the first time to participate in the CEOI, the USA team earned two gold medals (out of five awarded), one silver medal, and one bronze medal. Whether ranking by medal count or points, the USA's performance ranked highest among the 14 countries competing and surprised all observers.

Led by veteran Daniel Adkins (Baton Rouge, LA; MIT freshman), who missed overall first place by one point, the team included gold medal winner Matt Craighead (Eden Prairie, MN; 15-year-old high school senior), silver medal winner Adrian Sox (Ambler, PA; high school senior), and Bobby Knock (Houston, TX; high school senior). Both Adkins and Craighead scored perfect scores in one single day's competition.

Head coach Rob Kolstad was delighted. "Our guys really did well," he said. "These scores and this kind of tough competition should boost our performance in the November International Olympiad in Capetown, South Africa." Coach Greg Galperin (MIT graduate student) was also pleased. "The team's performance was outstanding," he echoed. "These guys have really worked hard and their scores reflect that dedication."

The USA Computing Olympiad team is sponsored by the USENIX Association. The team is coordinated by Don Piele, professor of mathematics at the University of Wisconsin-Parkside. The CEOI in Poland (for pre-college students) included four-person teams from:

- Belarus
- Croatia
- Estonia
- Germany
- Holland (only two contestants)
- Hungary
- Latvia
- Lithuania
- Poland
- Romania
- Slovakia
- Ukraine
- Yugoslavia

The CEOI is widely regarded as the most challenging computing olympiad outside the International Olympiad on Informatics. Two five-hour sessions require creating algorithms to solve a variety of problems. Solutions are then graded for correctness and speed.

For more information, see the Web page at <http://usaco.wup.edu> and subscribe to the hs-computing mailing list by sending a 'subscribe hs-computing' message to <majordomo@delos.com>.
Twenty Years Ago in UNIX NEWS (a.k.a. ;login:)

I intend over the next months (years?) to chronicle the growth of the USENIX Association and its publications in retrospective articles concerning “20 years ago,” beginning with 1977.

1977 was an important year in the history of USENIX and of UNIX in general. The newsletter was called UNIX NEWS at that time, edited by Mel Ferenzt, then at Brooklyn College, and appearing on purple dittos, which faded over the years; I am grateful to Tom Ferrin, who had the foresight to copy his issues before they faded. UNIX NEWS appeared ten times a year; though issues were frequently late.

Volume 2, #1 (December 1976-January 1977) contained the “Yale Shell” written by John Levine (co-author of UNIX for Dummies and More UNIX for Dummies). The Yale shell was a “proper superset of the standard shell . . . [including] alpha and numeric shell variables as well as a private ‘bin’ directory for user shell command files which are accessible regardless of your current directory.” Note that at the UNIX Users’ Group meeting at Yale in October 1976, it was decided that the Yale shell “would be adopted as a Users’ Group ‘standard.’”

2.2 (February 1977) announced the summer meeting of the Users’ Group to be held at Urbana, IL, May 19-21, and a bug in ttyn(III); the fix (by George Rolf at the Catholic University in Nijmegen) appeared in the May 1977 issue.

2.3 (March 1977) contained two letters from John Lions and the announcement of the availability of the V6 Code and Commentary volumes. Ferenzt wrote: “Ken Thompson has seen the first version of the book and says it is a good job.”

For nearly two decades this has been the most important computer book never published. I’ve seen fifth- and sixth-generation photocopies. It’s now available again. Ken Thompson says, “After 20 years, this is still the best exposition of the workings of a ‘real’ operating system.” (ISBN 1-57396-013-7. $29.95 in the US. Peer-to-Peer Communications.)

One of the Lions letters concerned the book (A$10.00 + A$7.70 air postage; about US$19.50 at that time); the other announced, “A second meeting of UNIX users in Australia was held on February 18th, at the University of New South Wales, with an attendance of about 30 people.”

2.4 contained the program for the Urbana meeting and a map of the University of Illinois campus. The May–June (2.5) issue noted, “The Urbana meeting was attended by over 150 people and was a great success . . . . At the meeting it was said (announced a too strong a verb) that Bell is preparing Programmers Work Bench for release this summer with Version 7 of UNIX soon thereafter.” There was also the bug fix from George Rolf and a note about two bugs in creat and their solution from George Goble at Purdue.

That May–June 1977 issue of UNIX NEWS was its last. Beginning in July 1977, the publication was called ;login:. Mel Ferenzt had been phoned by an AT&T lawyer and told that the group (it still had no name) could not use the term UNIX, because they had no permission to do so from Western Electric. At the meeting of UNIX USERS at the College of Physicians and Surgeons, Columbia University, May 24-27, 1978, the name of the publication was changed.
Also in July (2.6), *login* appeared in a new format: as Ferentz put it, "CUNY is now heavily into CROFF, new version." In September (2.7), PWB was announced as available; Ferentz also noted "A paper by J.A. Dolotta and R.C. Haight... describes the system and may be requested of Ted Dolotta."

*login* for October 1977 (2.9) carried the note that "The membership in the Users' Group now exceeds 250," Ken Thompson and Dennis Ritchie had given the first UNIX paper in October 1973; the paper was published in *CACM* in July 1974; even before that, the first Users' Group meeting (May 15, 1974) drew two dozen people from a dozen institutions. The June 18, 1975, meeting had "over 40 people from 20 institutions."

As you reach the end of this fragment, remember the Bell System's policy:

- no advertising
- no support
- no bug fixes
- payment in advance

No wonder the brave users needed a support group.

### Deadline for Student Proposals

The next deadline for submitting proposals for student scholarships, research grants, and undergraduate software projects is December 1, 1997. Details about each of these programs and guidelines for developing proposals are on the USENIX Web site at [http://www.usenix.org/students](http://www.usenix.org/students). Proposals should be submitted to Ellie Young, USENIX Executive Director.

Notification of acceptance will occur on December 20.

Proposals for these student program are considered on a quarterly schedule. Here are the upcoming deadlines for the rest of 1997 and the first half of 1998:

<table>
<thead>
<tr>
<th>Proposal Submission Deadline</th>
<th>Notification of Acceptance Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 1, 1998</td>
<td>March 31, 1998</td>
</tr>
<tr>
<td>June 1, 1998</td>
<td>June 30, 1998</td>
</tr>
</tbody>
</table>

### 1998 USENIX Nominating Committee

by Ellie Young

By Ellie Young

The biennial elections of the Association's Board of Directors will be held in the Spring of 1998. At press time, the current Board was seeking someone to chair a committee to nominate candidates for the Board. By the time you receive this issue of *login*: the nominating committee and chair will have been announced on [comp.org.usenix](http://comp.org.usenix) and on the USENIX Web site.

Board members are elected for two years and will take office no later than July 1, 1998. There are eight board positions:

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

All positions are typically filled from a combination of continuing Board members as well as new folks.

Suggestions for nominees will be accepted until early December. Please email them to: <nominate@usenix.org>.

### USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

- **Free subscription to *login***, the Association's magazine, published six to eight times a year, featuring technical articles, system administration tips and techniques, practical columns on Perl, Java, and C++, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

- **Access to papers** from the USENIX Conferences starting with 1993, via the USENIX Online Library on the World Wide Web: [http://www.usenix.org](http://www.usenix.org).

- **Discounts on registration fees** for all USENIX Conferences, as many as eight every year.

- **Discounts** on the purchase of proceedings and CD-ROMS from USENIX conferences.

- **PGP Key Signing service** available at conferences.

- **Discounts** on BSDI, Inc. products. BSDI information: 800 800 4BSD.

- **Discounts** on all publications and software from Prime Time Freeware, including Prime Time Freeware for UNIX, Prime Time Freeware for AI, Prime Time TeXcetra, and Tools & Toys for UNIXware.

- **Discounts** on all publications from The Open Group.

- **Savings** 20% savings on all titles from Prentice Hall PTR (<http://www.bookpool.com>) and from O'Reilly & Associates (800 958 9938).

- **Savings** (10-20%) on selected titles from McGraw-Hill (212 512 2000), The MIT Press (800 356 0343), Prentice Hall (201 850 6789), Morgan Kaufmann (800 745 7323), and Sage Science Press (805 499 9774)

- **Special subscription rate** to *The Linux Journal* (206 527 3385)

- **The right to vote** on matters affecting the Association, its bylaws, election of its directors and officers.

- **Optional membership** in SAGE, the System Administrators Guild

For information regarding membership or benefits, please contact <office@usenix.org>

Phone: 510 528 8649
Seeking Volunteer Conference Photographers

Help sklogis capture conferences in action! If you’re an experienced photographer and are planning to attend an upcoming USENIX conference, we’d greatly appreciate any photos you could provide us. Specifically, shots of keynote speakers and other luminaries, and perspectives on crowds of attendees that convey the atmosphere of the conference, are desirable.

Anyone who wishes to provide photos of the Conference on Domain-Specific Languages (October 15-17), LISA ’97 (October 26-31), the USENIX Symposium on Internet Technologies and Systems (December 8-11), or the 7th USENIX Security Symposium (January 25-29, 1998) should contact Eileen Cohen, sklogis: Managing Editor, <cohen@usenix.org>, before the event.

Call for Tutorial Proposals

by Dan Klein
USENIX Tutorials Coordinator
<dklein@usenix.org>

In an effort to continue to provide the best possible tutorials to the technical community which is served, the USENIX Association is soliciting proposals for future new tutorials at its conferences. The tutorial proposals may cover any subject, ranging from introductory to advanced materials.

The type of tutorial we are most interested in are introductory or overview tutorials for advanced people. We tend to avoid overly introductory materials (i.e., a proposal on “Introduction to C Programming” would not be appropriate). Previous conferences have included tutorials on such diverse topics as UNIX Network Programming, Java Security, Topics in System Administration, Multi-threaded Programming, Kernel Internals, Performance Tuning and Monitoring, and Software Contracts and Intellectual Property, among many others. Tutorial instructors are paid for their presentations, have their travel and reasonable expenses reimbursed, and receive a complimentary conference registration.

Tutorials usually run for a full day (six hours of class time plus morning, lunch, and afternoon breaks), although the smaller symposia and the LISA conference also have half day (three hour) tutorials. A proposal should include a statement of what you want to teach, and a coherent outline to your tutorial (not simply a list of what you want to cover, but the order in which you want to cover it, with an estimate on the amount of time for each subject).

Because a full-day tutorial lasts on the order of six hours, we need to know that you can comfortably fill that time, but not seriously overfill it (i.e., that you won’t suddenly discover at 4:30 that you have another three hours of slides left to present). If you have any supplementary materials to distribute (e.g., copies of papers, shell scripts, source code, illustrations, etc.), give an indication of the volume of supplementary material, and a rough count of the number of slides you will be presenting during class.

Historically, a typical tutorial has between 75-200 slides, along with up to 200 pages of supplementary material. If possible, include a couple of sample slides (one with text, one with a graphic) with your proposal.

If you have a complete or draft course already done, having a copy of the current materials available would be most useful.

We also need to know if you will be presenting or distributing any source code. If so, is it copyrighted by someone other than you? If you do not hold the copyright, you must be able to demonstrate that you have permission to use this material (we want to avoid requiring course attendees to have a source license).

Because the USENIX tutorials fall outside of the “fair use” clause of the US copyright law, the same rules apply for supplementary papers or reports.

Finally, your proposal should also include a summary of your previous teaching or lecturing experience, as well as a couple of references (that is, one or two people who have seen you teach that we can contact). These may be your students, supervisors, or colleagues.

Remember, we are looking for a proposal so nothing you submit will be cast in concrete. You may later decide to change some ordering of materials, or we may suggest some changes. You needn’t worry about getting it perfect the first time around. What we are trying to do is get a very solid feel for what you are offering.

The tutorial schedule for all conferences is usually scheduled four to six months in advance of the conference – the earlier we receive a proposal, the more conferences it can be considered for. Please send your proposals to <dklein@usenix.org>, or by physical mail to:

Daniel Klein
USENIX Tutorial Coordinator
5606 Northumberland
Pittsburgh, PA 15217-1238

Be sure to include an electronic and physical address and a phone number. All proposals will be acknowledged upon receipt.
Strange Bedfellows

by Tina Darmohray

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

I’ve never been an Apple fan. That’s not to say that I have tried them and don’t like them; it’s just that I can take them or leave them, so I’m not a “fan.” It’s been my observation, though, that the PC of choice for many UNIX users is an Apple, and they’re often pretty fanatic about it.

In fact, when I shopped around a few years ago for a laptop to run the common shrink-wrapped financial software, an awful lot of my friends really pushed hard for an Apple product. I wound up opting for a cheap PC because I could get more machine for the same amount of money. My financial software works fine on my PC, save the occasional PC/OS crash under heavy labor.

Despite my lukewarm feelings about their products, I’ve followed Apple’s recent business-roller-coaster ride pretty closely. But I really wasn’t prepared for the announcement that Microsoft would invest in Apple! I mean, I guess I knew it was a possibility, and certainly a much-repeated, off-color joke by the anti-Microsoft crowd, but that’s a long way from really thinking it would happen. But it did. So now I find myself wondering what the Microsoft investment means for system administrators, other computing vendors, tech stock investors, school kids, legal precedents, and whatever else it might affect.

I have to confess that the message isn’t clear to me. I guess that’s not too surprising, because the message isn’t clear to many who are more astute than myself. So I decided to break it down to the very basics: to categorize the computer solutions into what they had in common, and where they differed, before I considered why some were winning in the marketplace — sort of an apples and oranges approach (please excuse the pun).

As I mentioned, when I chose my PC, I looked for the most machine for the least amount of money. Commodity PC hardware running a Microsoft OS with lots of available software won that shop-and-compare battle, hands down. It wasn’t until just recently, when I was attempting to put multiple add-on cards in a PC and found that Plug-and-Play doesn’t, that I realized how valuable a complete, integrated solution is. It was then that I decided that, at first glance, commodity hardware may seem cost-effective, but there is a cost (in my case, two days of my time) in self-integrating. From that experience, I decided that Apple is more of a competitor with total-solution vendors, like Sun and SGI, that sell an operating system expressly for their hardware, than it is with a software-only supplier, like Microsoft. Based on that analogy, Sun, SGI, HP, DEC, and Apple, are all competitors.

In contrast, it seems to me that Microsoft is a strictly software company that sells computer operating systems and applications. They sell the software, and you buy the hardware anywhere you choose. I can think of just a few competitors in that arena: Sun and BSDI come to mind. And if you get legalistic about it, you’re probably down to BSDI. I’m sure I’m overlooking companies in both categories, but you get my point: maybe, strictly speaking, Apple isn’t really Microsoft’s competitor after all, despite our tendency to group them because they’ve both been viewed as low-end, personal computing solutions.

Tonight a consultant friend of mine called. I asked his opinion of the Microsoft news. He said he was stunned and confused by it. He said he’d heard different versions of what had actually transpired. I tried to clarify it based on the business articles I’d been reading. He responded with something like, “So, let me get this straight: Microsoft invests money in the hardware vendor that currently represents the least threat, makes money in doing so, and avoids the antitrust lawsuit thing all for $150 million? Brilliant move.”

I think someone should tell the monopoly police that Microsoft gave the money to the wrong “competitor.”

A Year of Change

by Hal Miller

Hal Miller is president of the SAGE STG Executive Committee.

It has been a year of change and new direction. For the first time, SAGE sponsored a tutorial session for high school student sysadmins: the Maryland Virtual High School project was a two-day training session for 22 students from all over that state.

For the first time, SAGE co-sponsored a non-UNIX conference: the Sysadmin of NT Workshop in Seattle, which backed up to the USENIX Windows NT Workshop. These events brought together UNIX sysadmins, NT administrators, and Microsoft for the beginning of what is to become a series of gatherings and technical exchanges.
For the first time, SAGE published a booklet designed to be read by an audience other than system administrators: the Short Topics series now includes *System Security: A Management Perspective*.

The SAGE Board election process and timing have been changed to match that of the USENIX Board. The timing of LISA has changed slightly to keep it half a year opposite the USENIX General Conference as that one moves to June. The Web pages look different. We have reviewed the various drafts of articles of organization and policies and installed a complete set of documents (available on the Web) for the first time.

We have a new Webmaster (shared with USENIX) in Peter Collinson. Referring back to my column in the August issue, where I discussed the extension of board seats and mentioned that Paul Evans would be ending his term without extending, we welcome back Kim Trudel to fill that extra period. Again, our gratitude to Paul for all that he has done and a personal “warning” to him that I won’t let him get too far away. Even the name “SAGE Board” has been replaced by “STG Executive Committee” to keep in line with changes in USENIX documents.

At the same time, it has been a year of continuity. LISA has returned to San Diego (appropriate facilities, decent price for the size of the conference). We have again assisted in getting the SANS conference underway. *Job Descriptions for System Administrators* is out in its second printing, and new booklets are about to hit the street. The SAGE share of *login: continues to grow, as do the membership numbers. Local groups are continuing to flourish. The list of things to do is still very long.

It is time to start setting our goals for the coming year. We are looking to start more booklets into the publication pipeline. We are looking at putting together another conference for very large site administration (alternately dubbed “real LISA” or “teraLISA”). We continue to investigate the issues involved in education and certification. We expect to increase our level of international cooperation and our local group participation and support. We are working on a “recommended toolkit,” a speakers bureau, and new ways to recognize top sysadmins.

Let us know what you think. It’s the only way we can ensure that SAGE does and becomes what you want.

---

**SAGE STG EXECUTIVE COMMITTEE**

**President:**
Hal Miller <halm@usenix.org>

**Secretary:**
Tim Gassaway <gassaway@usenix.org>

**Treasurer:**
Barb Diiker <barb@usenix.org>

**Members:**
Helen Harrison <helen@usenix.org>
Amy Keeling <amy@usenix.org>
Paul Evans <ple@usenix.org>
Pat Wilson <paw@usenix.org>

---

On July 18-19, Dr. Evi Nemeth, author of *The Unix System Administration Handbook*, presented a two-day seminar to student system operators (sysops) who are part of the Maryland Virtual High School Program. USENIX and SAGE contributed financial and organizational support to the seminar.

The Virtual High School is a network of 15 Maryland high schools linked by the Internet to share information and resources in computational science projects. Sysops from each of the participating high schools had an opportunity to attend the seminar and to share their expertise with classmates and faculty when they returned to school in the fall. In addition, visiting staff and students from Arlington County, Virginia joined in. Dr. Nemeth was assisted by University of Colorado student Adam Boggs.

Dr. Nemeth, who is an associate professor in the computer science department at the University of Colorado and a former USENIX board member, focused the seminar on UNIX system administration including hardware and software, file sys
tems, Web configuration, security, and issues of privacy and ethics related to computer network use. The two-day event also included an address by Andrew Humé, Research Scientist, ATT Labs Research, and president of USENIX, who talked about using computers to solve real-world problems involving large amounts of data. According to Nemeth, “Andrew’s talk on big data problems was great. [The students] were saturated by that point, but welcomed the change and seemed to really enjoy it.”

The Virtual High School program is funded by a three-year $1.5 million grant, awarded in 1995, from the National Science Foundation. It brings to the classroom a team approach to problem solving and a technology-rich environment that mirrors scientific investigation currently used in research and business. The program is coordinated statewide by Montgomery Blair High School, which was the venue for the seminar. A temporary all-Linux lab at Blair was set up and administered by Matt Shiblea, MVHS Network Coordinator.

Mary Ellen Verona, project director for MVHS, called the seminar an unusual opportunity for students to receive training from top-notch professionals. “[W]e have taught UNIX basics, but could not compare with the wealth of background, and especially the style of presentation. It’s that ‘at the tip of your fingers’ knowledge that shone through — there was no way that the kids could ‘get too far ahead’ — because a new fine point, anecdote, or illustration was always forthcoming.” Verona also believes that this occasion will help demonstrate the ability and value of student sysops to district technical leaders. Montgomery Blair High School and other project schools serve as seed sites in expanding understanding of networking technology for school use throughout Maryland.

More information about MVHS can be found at <http://mvhs1.mhhs.edu/mvhs.html>.

Evi Nemeth holds forth

A Thousand Pints of Lite

by Pat Wilson

Pat Wilson is a member of the SAGE STG Executive Committee. She’s an Invited Talks Coordinator for USENIX.

<paw@usenix.org>

Today’s rant concerns machines. Lots of machines. Now, I don’t know about you, but here in academia we’re loathe to part with “obsolete” hardware, especially if it still can be considered to work. Lately we’ve refreshed the workstation fleet, so we’ve got old RS/6000 320s and early DEC Alpha boxes around, and (for other reasons) lots of spare disks. Of course, if you give these machines away to deserving grad students, you’re just buying yourself a support nightmare, and there are all these projects to do...

As I write this, I’m slapping an OS on yet another RS/6000. This one will be the SAMBA (and netatalk, if I can figure out how to run it on AIX) sandbox. I’ve got old RS/6000s doing nameserver duty, running the AFS cell, testing ADSM, running dialup authentication, and acting as
Web servers (though we really prefer to use ancient DEC hardware for that). It's quite convenient – until someone finds a gaping security hole in our standard software set and all of the machines need to be upgraded. At last count, I have primary or secondary responsibility for over 30 servers of various natures, and that number continues to grow (there are already 2 more in the pipeline).

I know that I could stop the madness by being a little more intelligent about what I run where – or could I? When reasonably powerful hardware is so cheap (essentially free) and demand for new and better services is constant, it's not clear to me what to measure "economy of scale" against. More machines theoretically mean better redundancy and certainly mean that the users/clients aren't complaining about some other group's application interfering with their work. I can also have the luxury of keeping a new server hidden away until I'm happy with it, which would be more difficult on a box that was already visible. And, of course, there are many fewer repercussions to rebooting a machine that does only one thing.

Perhaps the practical limiting factor will wind up being the size of my machine room – the current "server motel" (servers check in, but they don't check out) is filling up fast!

**Eewwie GUI!**

I ran across a new ad campaign last week. "If network managers wanted to write scripts," the copy reads, "they'd all move to Hollywood." Besides the obvious impracticalities of such a notion (if all the "network managers" were in Hollywood, it might be quite hard to adequately maintain a worldwide network, and the lines at any caffeine outlets would be hellish), this gave me an excuse to launch into my favorite rant: the evils of binary-only GUI tools.

Have you ever found a vendor-supplied GUI tool that did exactly what you wanted? Or are there always sticking points? Is it even possible to write a tool for (system|user|network) administration that works reasonably well at a variety of sites? I've certainly never found an adduser "point & click" interface that I could use out of the box and would do the right thing in my environment, much less make adding tens of users at a time painless (in the very real and physical sense). If the tool is supplied as a binary only, I wind up sticking bags fore and aft to make it work.

A more insidious problem, though, is the fact that people tend not to learn what's behind the happy fun icon. Go to any comp.* newsgroup today and you can find questions that make it patently obvious that these "sysadmins" have never really understood what it is that they're doing when they click on "partition the disk" or "add a tape drive." These folks aren't stupid, but they have been lulled into a false sense of security. When things go wrong (due to a bug in the GUI itself, underlying OS issues, or some local catastrophe), they've got no idea where to begin to look for the problem themselves. When bad assumptions are made by the GUI designers (it really does seem like a great idea to remove a user's $HOME when you're removing the user, but if you were doing it by hand and the user's home directory was /, you'd probably think twice about it, unlike the sysadmin GUI that didn't think to take this into account), they're often transparent until it's too late.

GUIs are supposed to be labor-saving productivity enhancers. They do enable folks who don't intend to be sysadmins to do routine tasks for themselves; they're also useful for those "one off" tasks that you don't have time to crawl through the man pages for. However, they do us, as professional sysadmins, a real disservice. Not only do they usually hide the mechanics of the task from the admin, but they can give the impression that it's possible to reduce the complexity of what the trained sysadmin does to a list of automated tasks.

GUI tools are like TV dinners. They're OK when you're in a hurry, but they're no substitute for home cookin'!

**Stretch Your Wardrobe at LISA**

Come to the costume party and reception at LISA in San Diego on Thursday, October 30, 6:00 pm-8:00 pm. You are encouraged to come in costume dressed as your favorite (or least favorite) system command, computing equipment, or high-tech concept! See <http://www.usenix.org/events/lisa87/> for the entire LISA program, including this and other social activities.
A System Administrator’s View of LDAP

Recently, here in Netscape’s IS group, we’ve been using LDAP ("Lightweight Directory Access Protocol") directory services on our internal network. This should not be surprising because Netscape’s server products and Netscape Communicator use LDAP for sharing information. Although LDAP will help developers and end-users, systems administrators may reap the greatest benefits from LDAP deployment. However, to reap those benefits, system administrators will want to have a good understanding of what LDAP can and can’t accomplish, become familiar with LDAP basics, and then begin to transition to directory-enabled utilities.

Why LDAP?

At a recent BayLISA meeting, LDAP was being correctly described as a system for distributing information like lists of users. A concerned sysadmin was quick to ask, "With all of the user lists that I manage already, why do I want to have another one?" A fair question, I thought, and one that hints at why system administrators should pay close attention to LDAP’s development.

LDAP has the potential to replace existing application-specific lists and consolidate information. This means that changes made on an LDAP server will take effect for every directory-enabled application that uses this information. Imagine adding a variety of information about a new user through a single interface once. Immediately, the user will have a UNIX account, NT account, mail address and aliases, membership in departmental mailing lists, access to a restricted Web server, inclusion in job-specific restricted newsgroups, and be included in the company’s phone list, mail address book, and meeting calendar system. When a user leaves, access can be disabled for all of these services, again from a single operation.

That sounds wonderful, but at the same time far-fetched. How can we expect operating systems and applications from different vendors to agree on one system for looking up information, and why would LDAP be perceived as the key to making this possible?

First, let’s look at what LDAP is and isn’t. Developed at the University of Michigan, LDAP is now an Internet standard for directory services that run over TCP/IP. One or more LDAP servers contain the data that make up the LDAP directory tree. An LDAP client connects to an LDAP server and submits a query to request information or submits information to be updated. If access rights for the client are granted, the server responds with an answer or possibly with a referral to another LDAP server where the client can have the query serviced.

An LDAP server is not simply a form of database, but a specialized server for directories. A directory can be distinguished from a general-purpose database by the usage pattern. A directory contains information that is often searched, but rarely modified. Hostnames or usernames, for example, are assigned once and then looked up thousands of times. LDAP servers are tuned for this type of usage, whereas relational databases are much more geared to maintaining data that are constantly changing.

Another difference is that relational databases store information in rows of tables,
whereas LDAP uses object-oriented hierarchies of entries. One could think of this hierarchy as being similar to DNS. Because an LDAP directory could hold host name information, it might seem that LDAP could be a replacement for DNS. However, DNS is very specialized, and LDAP was not designed to address the same set of problems for which DNS has been groomed. Still, DNS maintenance can benefit from an LDAP-based strategy.

By designing for this usage pattern, current directory servers with a million or more entries can respond to hundreds of search requests per second from a single server. Replication is also possible, which makes LDAP very scalable.

Reducing load on the authoritative server is not the only reason for using replica servers. As with YP, putting replicas on subnets can avoid network traffic through routers and reduce latency. However, unlike YP, the synchronization scheme features incremental updates that can be pushed immediately to the replicas rather than periodically transferring all of the data. For Netscape's internal mail hubs, we are also experimenting with running replicas on the local host. Although this really isn't necessary, it does allow Netscape Messaging Server to do lookups locally with very little overhead to maintain synchronized data.

**LDAP Basics**

Before going any further with architecture issues, let's look at how LDAP organizes information. LDAP introduces a lot of new terminology, but the only terms you need to understand to get started are *entries*, *object classes*, *attributes*, and *distinguished names*.

An LDAP server contains *entries*. Each entry's type is defined by an *object class*. Object classes define which *attributes* are required and other optional attributes that can be associated with an entry of that class. Each entry is uniquely identified by a *distinguished name* or DN in LDAP parlance. These DNs are organized in a hierarchy.

For example, I am at a company in the United States. The top entry in the hierarchy for my entry has the DN "c=US". This entry is of the object class *country*. A country entry has one required attribute for c which has the value of US. "c=Netscape Communications Corp., c=US" is the DN for an entry of the object class *organization*. This requires the attribute o which in this entry is "Netscape Communications Corp." Notice that the parts of the DN are separated by a comma.

At this point, what we have would likely be the BaseDN for this server. A BaseDN defines the top of the namespace that the server is responsible for much like a DNS zone. As we add entries further down the hierarchy, the DNs will become longer. Remembering a long DN is not an issue for end-users because client applications will be doing the searches then displaying the attributes the user needs to see. For most applications, the full DN does not need to be exposed to the end-user.

Now that the groundwork has been laid, let's look at an entry for an individual.

"cn=Bruce Markey, o=Netscape Communications Corp., c=US"

This is of the object class *inetOrgPerson*, which requires a cn for the common name. It also requires an sn for the surname, which is an example of a required attribute that is not the attribute included in the DN. The *inetOrgPerson* class also defines about 50 other attributes that can be associated with a person, such as uid, title, manager, phone, pager, email address, and other information you would likely want to associate with a person in an organization that has access to the Internet. It even has attributes for jpegPhoto and audio, which are possible because attributes can be declared to be encoded into different *syntaxes*, including binary data.
In order to maintain authoritative information, access control needs to be imposed for privileges to read, write, search, or compare. Access control can be done on a subtree, entry, and/or attribute type and granted to individuals, groups, or “self,” which allows authenticated users to access their own entries. This allows a great deal of flexibility. For example, you might want to allow only people in a human resources group to change the title or manager attributes, allow administrative assistants to change office location and pager number information for just their department, and allow individuals to modify home phone number, license plate, etc. for their own entry.

**More Than a Protocol**

Understanding how to maintain the data alone is not enough to put LDAP to use. There are four well-defined pieces of the overall system that simplify implementing LDAP. The LDAP open standard, the API, the LDIF text format for data, and the object class definitions. Let’s look at why each of these is important.

**LDAP**

RFCs 1777 and 1778 define the protocol that enables clients from different developers on any platform to talk to any type of LDAP server. Many vendors have announced support for LDAP. Notables include Netscape, Microsoft, and Novell, each of which already offers directory-enabled servers and clients. The University of Michigan, where LDAP evolved, has source code for its original *slapd* server and other tools available for download.

**API**

One of the important factors for LDAP to succeed is that developers should be able make use of information from an LDAP server without having to write and debug a lot of code. A well-defined Application Programming Interface mitigates this problem. You may be familiar with SOCKS and *socksifying* an application. With LDAP, *directory enabling* is a very similar process. For a program that could make use of directory information, include the API libraries in the source directory, modify the program’s code to call the API functions at the point where the information needs to be looked up, and then recompile. The most recent version of sendmail already includes the API and has options to look up information through LDAP.

**LDIF**

Another piece of the puzzle that I’ve found to be remarkably important is the LDIF file format. This ASCII text format is used for exporting and importing data to and from LDAP servers. This not only makes it easy to migrate data from one server to another, but also allows you to write scripts to create LDIF files from other data sources. You can then verify and manipulate the LDIF file before committing the data to the server. Because command line tools like “ldapsearch” return data in LDIF format, you can save some or all of your data to a file, make global changes, then import the new data back into the server.

**Object Classes**

One other piece that is important to portability is object class definitions. If a client needs some attributes that aren’t in the well-known object class definitions, a new object class could be created as an extension of a similar object class. The client could then work with any LDAP server so long as the server has been given this new object class definition.
Transitioning to LDAP

Even with a solid understanding of LDAP concepts, switching to LDAP won't happen instantly. There are two distinct ways to put information to use. One is by using applications that are already directory enabled, and the other is to gateway the information from LDAP into a format used by existing applications.

Here at Netscape, we first used a corporate-wide phonebook that was available through a Web page. This was the beginning of creating an authoritative list of employees on our central Directory Server. Netscape Communicator now includes a window that can do native LDAP searches. Not only can that act as the corporate phonebook, but results can be selected to automatically address email messages.

Once the data are being stored and updated on an LDAP server, other applications can take advantage of this resource.

Many LDAP-based software distributions include command line tools for searching and modifying directory information. This makes it possible to use or manipulate information with simple shell scripts. There are also tools available for programming in Perl, Java, C, etc.

Let’s look at UNIX login information as an example of some possible transition strategies. Once attributes for users are stored in a directory server, one useful thing that can be done is to sync up usernames and passwords for multiple environments. We've implemented a system in-house where UNIX and NT account passwords are updated when passwords are changed through our Directory Server interface. This not only simplifies the change for users, but can help to reduce the chances of there being infrequently used accounts with forgotten passwords.

One of the more interesting gateways to date is “ypldap” by Luke Howard of Xedoc. This is an NIS (YP) server that uses LDAP instead of files to look up its information. It supports passwd and group maps along with the other commonly used YP maps. This approach allows using existing YP-based applications without needing to run a script for converting the data.

Eventually, even tools like “ypldap” may become unnecessary when operating systems include directory-enabled /bin/login. Most UNIX versions have a configuration file that tell the O/S where to look for password information. It should be relatively easy for vendors to add LDAP as one of the choices along with NIS and the local /etc/passwd file.

As you can see from this example, LDAP is not an all-or-nothing proposition. Once directory services are available, applications that use the data can evolve.

Further Reading


Specific information about the protocol is in the RFCs. 1777 defines Lightweight Directory Access Protocol and 1778 details The String Representation of Standard Attribute Syntaxes.

ToolMan Edits Pipes – Interactively!

This issue I’ll describe a handy little tool that I use in many situations in my day-to-day work. This one should be a welcome addition to the toolbox of any UNIX shell hackers out there. It’s called vip (VI Pipe), and it enables you to interactively edit any point in a pipeline, whether at the beginning, middle, or end.

Toolz R Us

Normally, in a pipeline, when you need to edit some phase of the data stream, you use a standard tool such as sed, grep, or awk to alter, filter, or otherwise manipulate the stream. One potential problem with this approach is that the manipulations have to be very well thought out in advance. Another is that the manipulations will probably need to be applied uniformly. And third, the data must be very well understood in advance. Not all situations and data easily conform to these constraints.

Alternatively, when the changes needed for the data are more than trivial, or perhaps you just don’t feel like expending the mental energy needed to work out all the expressions in advance, a typical approach might be to run some process or pipeline, dump output to a file, edit the file with vi, pico, or emacs, then push the data along to the next phase by using the file as input to some additional process or pipeline. The catch here – other than the sheer awkwardness of this process – is that you have to remember to come back later and clean up all of those little and not-so-little “temporary” files.

So, wouldn’t you just like to be able to tap in an edit session at any arbitrary point in the pipeline, do your magic on the data, then have it automagically continue on its merry way? The vip program provides this functionality, and operates syntactically just like any other filter.

Example 1

Here’s a little example. Suppose you want to send Xena a zephyrgram. (Perhaps you’re working from a Telnet session, and xwrite is not an option.)

Option 1: zwrite xena. You have to type in lines of text, and once you hit “<return>” on a line, you can’t go back and fix things. Goof up badly enough, and all you can do is hit “<control-C>” and start all over.

Option 2: Edit a file, then zwrite xena < tfile. Then you have to remove the file. This is tedious.

Option 3: vip -n | zwrite xena. Edit a message, exit the editor, and away goes the zgram, no temp files to worry about. (The -n tells vip not to read standard input before invoking the editor.)

Example 2

Let’s make it a little more complicated. Let’s say that part of what you want to zgram Xena about is a report of disk utilization produced by the df command. In this case, you could type something like

df -lk | vip | zwrite xena

When the editor starts up, df’s output is already there in the edit buffer. Add your embellishments and exit.
The “ToolMan” articles, seen in recent issues of *login*, will continue on a regular basis. ToolMan needs your help! If you have written a software tool that is unique, useful, or really cool, please send ToolMan <des@cs.duke.edu> a description, and he'll try to feature it in an upcoming issue.

**Example 3**

One more example, this time with vip at the end of a pipeline. Suppose we have some subdirectories and want to copy a new version of a file into a few of them. The subdirectories look like this:

```bash
$ ls -F
Distfile
  sgi-IRIX64_6.1/  
  dec-OSF1_V3.2/ 
  sparc-SunOS_5.3.9/  
  1386-SunOS_5.4@ 
  1386-SunOS_5.5@ 
  sparc-SunOS_4.1.3C@ 
  1386-SunOS_5.5.1/ 
  sparc-SunOS_4.1.3.U1G 
  sparc-SunOS_5.5.1/ 
  sparc-SunOS_4.1.4/ 
```

Run the command `ls | vip -o`. (The `-o` tells vip not to send the buffer to standard output after editing.) In the edit buffer, we edit the list and add some shell syntax to get this:

```bash
for DIR in \
  sgi-IRIX64_6.1 \n  dec-OSF1_V3.2 \n  sparc-SunOS_5.5.1 \n  1386-SunOS_5.5.1 \n  sparc-SunOS_4.1.4 
do
  cp /cs/puma/src/bin/zippy $DIR/bin
done
```

Then we send it to a shell with `":w !sh<enter>"` to execute the commands. If we decide to perform some additional operations on these files, such as to change their permissions, it's easy enough to just alter the `cp` line in the edit buffer and send the commands to a shell again. (This could have been done with `ls | vip | sh`, but that wouldn't have allowed for iterations of command executions.)

**Smoke and Mirrors**

I have to admit that temporary files are at work here. But the cleanup is automatic and transparent to the user. Also, just to prove that I'm not a vi nazi, vip is not restricted to just that editor. It will check your VISUAL and EDITOR environment variables and behave accordingly.

vip is a Bourne shell script that does some redirection, uses `umask` to protect privacy, and uses `tr ap` to clean up if a common signal is caught. It is, in fact, rather elementary.

**Cop It**

Even if you thought `cat` didn't have the stuff (August 1997), or you said to heck with `check` (June, '97), I think you'll like vip. The adventurous — and even the timid — can pick up a copy at `<http://www.cs.duke.edu/~des/scripts.html>` or `<ftp://ftp.cs.duke.edu/pub/des/scripts/>`. Don't forget the man page! Happy plumbing!

Note: A much faster version of `cat` (1.16 or later) is now available (and no more temporary sort files!).
On Reliability – Networks and Services

This issue’s “reliability” article is concerned with network and network service planning. This includes such things as routers, switches, cabling, leased lines, and servers for such things as mail, DNS, and file service.

Once again, it is important to understand what you want to accomplish before you set out to accomplish it: recall the ideas of service levels, risk evaluation, costs of failures, etc. (and if you don’t recall them, you may wish to see the articles in the June and August issues). In this article, I’ll point out a few places that tend to be “single points of failure” and try to suggest ways to deal with them. And remember, if reliability is to increase, so are cost and complexity – work to find the balance that is best for your organization.

Network Topology and Components
Let’s talk a little bit about network topology – the physical layout of the cables, routers, hubs, etc. that make up the “skeleton” of your network (if the main part of a network is the “backbone,” I figure that it’s fair to call the whole shebang the “skeleton”).

If you’re a small organization and everyone fits on one floor of a typical office building, your layout is likely going to be pretty straightforward. All the routing and repeating hardware will end up in your single wiring closet (which, for this size of network, may actually be a closet), with a single connection to your internet service provider (ISP). While you don’t have a lot of reliability choices in this instance, there are a number of things that you can do to help you recover quickly when a failure happens, and many of these tactics can be applied to local hubs in much larger networks.

Office Wiring Drops
I have three suggestions for your office drops: use quality components (i.e., “Cat 5” wiring, terminations, punch downs, etc.), install them carefully and within specifications (i.e., length limits, etc.), and install spares, because some wires and connectors will eventually fail, and it’s much nicer when you can just switch people to the next port to get them working again. You should also be careful where cables are run in the offices or cubicles. This is not just a safety thing; you don’t want people walking on or tripping over your nice, new cables either. You should probably consider hiring an outside contractor to do your wiring instead of trying to do it yourself. It will probably get done faster, and better, and you’ll have a written guarantee and someone else to point the finger at when things don’t work.

Local Hubs and Repeaters
Whether you choose fancy-shmancy smart hubs with all sorts of whiz-bang remote management features, or the dumbest, plainest hubs you can find to minimize the number of things that can go wrong, try to:

- standardize on one model or brand
- maintain a good relationship with your supplier
- keep a spare or two on hand

Make sure you have spare ports available in case a single port or group of ports goes bad. If your main file/mail/Web/doom server is the only fast Ethernet device you have, make sure you have a spare fast Ethernet port to use if the one in use goes bad. (If your

by John Sellens

John Sellens has recently joined the Network Engineering group at UUNET Canada in Toronto, after 11 years as a system administrator and project leader at the University of Waterloo.

<jsellens@uunet.ca>
organization is like most, you'll need extra repeater ports eventually anyway. Just remember to buy more repeaters when you start running out of ports!) Cascading hubs, where a number of slave modules are daisy chained off a single master, can be attractive if you need managed hubs, but make sure you're not left completely dead in the water when the master module fails.

Local Routers

If you're a small office, you may have only a single network, and your only router may be for your (single) connection to the outside world via your ISP. Routers tend to be relatively expensive, so it may not be financially practical to keep a spare on hand. If you have only one or a few routers, see if you can strike a deal with your ISP or router vendor (and you do have only one router vendor, don't you?) for quick replacement in case of failure, or, at the very least, purchase a maintenance contract that guarantees you overnight (or faster) delivery of a replacement. If not, be prepared to be cut off from the rest of the world at some point.

That pretty much covers local network design, other than to mention that you should consider the security and safety of your wiring closet. Ideally, you would like one that is 100% secure from prying eyes and fingers, is air-conditioned and rodent-free, has uninterruptible power, and is not located in a hurricane- or tornado-prone area, on a floodplain, or underneath a kitchen or washroom. Unless you're very lucky, you'll probably have to settle for something less than ideal.

The "upstream" or "backbone" portion of your network is where you will probably be more concerned about higher levels of reliability. It's one thing for a workgroup LAN to go down and isolate 15 or 20 people, but it's a different matter entirely when your global corporate backbone melts down and thousands of employees are left idle. This is where you should consider redundant paths and cables, uninterruptible power and good environmental controls, high reliability hardware, and very careful configuration and monitoring.

One of the most obvious reliability approaches in network topology is the use of redundant communication paths to guard against natural or backhoe-related cable failures. This is often more important when your organization is spread across multiple buildings, cities, or continents than when you are within a single building. Cable or fiber failures within a building are usually easier to find and deal with than a failure on a leased fiber somewhere between Chicago and New York. By building some sort of "looping" into your network (e.g., three buildings and each has a direct connection to each of the others), you can live with the failure of any one wide-area link, albeit at a reduced total bandwidth. When possible, you should consider the use of multiple access points to your building and the use of different wide-area communication carriers to reduce the risk of a single incident (fire, backhoe, disgruntled telco employee) taking out both of your redundant links. For example, a power outage at a communication carrier can be a big problem for your network if your only connection goes through that switching office.

One alternative you might want to consider instead of private leased lines is the use of the Internet for wide-area communication links, using virtual private networks (VPNs can be implemented using software or hardware encryption to provide secure communication over public paths). The reliability advantage is that you can use the multiple redundant links of your ISP (and the rest of the Internet) to provide a reliable communication path. This, of course, can also be a disadvantage, because you're depending on someone else to provide appropriately reliable service for your network.
The routing and switching hardware you use on your WAN is also a very important part of reliability. The unfortunate thing is that, as your network grows larger, it often makes the most sense (from a bandwidth and management point of view) to use larger routers, rather than collections of smaller routers. This means that the cost of a router goes up, as does the cost of keeping a spare available. But fortunately, the larger routers tend to be more reliable, with such things as redundant power supplies (make sure they're on different building power circuits!) and multiple, independent interfaces. In practice, you're more likely to have a cabling or WAN circuit problem than a router problem (or at least a router problem that can't be fixed with a more current software release or a reboot).

And, as always, label, document, and be ready for problems. Make sure that each cable is labelled (with something that won't fall off!), make maps of floor plans and network drops, map your network links, and keep your vendor support numbers handy, with a list of part numbers, options, and wide-area communication circuit numbers. Make sure you have more than one copy of your documentation, including a paper copy, in different locations so you won't be unable to reach the copy you need to recover from a fire or natural (or network) disaster. (This, of course, includes the configurations for your routers. Don't keep your only copy in the router's memory!) Remember to plan ahead to limit problems in the first place and to make it easy to recover when you've had a failure.

**Network Servers**

Once you have your physical network in place, you may actually want to put it to use by connecting some machines. I'll claim that, other than the networking infrastructure, you can pretty much split network-connected machines into clients and servers (though it is never actually that clear-cut in practice). I'll define a "client" as a machine that no other machines or services rely on. A client machine is one that can disappear off the network and the only people inconvenienced will be those who want to sign on to that machine directly (e.g., personal workstations). I'm going to ignore client systems in this article, see the August issue for my comments on computing hardware reliability.

I think it's worthwhile to differentiate between "servers" and "services." The server is the hardware; the service is the protocol or information that the server provides or records. Increased reliability is easier if your service can be decoupled from your server (i.e., if the service in question does not require special purpose hardware and replicates easily, you're in luck). Examples of services that decouple and replicate well are DNS name service, DHCP and BOOTP service, and (most) Web servers; services that don't fare as well are file servers, database servers, and mail servers (because you usually want to have one "authoritative" server for these purposes).

For those services that don't decouple or replicate well, there are two basic approaches to reliability: make the server machine itself as reliable as possible (see the August article) and/or make it as easy as possible to move the service to a different machine in the event of a failure. The latter approach is more or less practical, depending on the service and the size of the data and client populations. But planning, record keeping, and preparing will make service movement much easier.

One technique that is applicable when you have multiple networks is the use of multiple network interfaces on your servers. For example, if you have a file server with a separate network interface for each network that it serves, router failures won't interrupt your file service. This technique can, of course, be applied to just about any service and will usually provide better performance as well as better reliability.
For those services that can be replicated, the obvious approach to reliability (and often performance) is to replicate. For example, DNS service is usually best provided by a primary server and multiple secondaries, geographically dispersed throughout your company. That way, a failure (power, network, or human) that isolates the part of your network containing your primary nameserver won’t disrupt your other networks (unless the failure continues long enough that the DNS data start to time out). Topological (physical and logical) dispersion is a very important technique on nontrivial networks. Other services that benefit from replication and dispersion are security servers (e.g., kerberos or SecurID), relatively static file servers (e.g., software servers for your workstations), etc.

For other services, such as mail and USENET news, replication usually isn’t appropriate (or just plain doesn’t work). However, depending on the size of your organization, you should consider having multiple servers, with people assigned to servers based on location or some arbitrary differentiation such as user id. This is the “don’t put all your eggs in one basket” reliability technique. It doesn’t help those people assigned to the failed server, but at least the people on other servers can keep on working.

One alternative that can be especially appropriate for Web servers is to co-locate your Web server with your ISP or other service provider. This means that your Web server is no longer limited by the bandwidth or reliability of your network link to your ISP, and you can benefit from the UPS, air conditioning, or 7x24 services from your ISP without having to do it all yourself.

Software and Configuration
In addition to the network and system hardware and planning, proper configuration and software support are essential for a reliable network and services. The following are some useful techniques.

Automate Your Configurations
This makes it easier to maintain and replicate your servers and services and makes it much less likely that a finger slip will cause your servers to stop working. One of the most obvious places for automation is when configuring your routers, especially security-related routers (see [1] for an example).

Use a Dynamic or Load-balancing Nameserver
Use a dynamic or load-balancing nameserver (such as “lbnamed” [2]) so that DNS lookups will ignore redundant servers that are down or otherwise unavailable. There are also hardware devices, primarily sold as gateways to multiple WWW servers that serve the same URLs, that act as gateways to the private server network and that choose the fastest or currently available server.

Configure Your Client Machines (When Possible) Using Protocols like BOOTP or DHCP
Properly configured, a simple config file change can cause all the client machines in your organization to choose different servers (e.g., DNS, time, gateways, etc.) the next time they boot, which makes it much easier to reconfigure things in case of a failure.

Use DNS MX Records for Your Mail Machines
Use DNS MX records for your mail machines to cause incoming mail to collect on another one of your servers when a mail machine is unavailable. This is much more convenient than having your mail pile up on the sending system because it allows you to adjust the expiry time or manually redirect the mail elsewhere to accommodate the failure and provide alternative recovery methods.
Use DNS CNAME (Alias) Records

Use DNS CNAME (alias) records to name your server machines so that it's easy to move the service to another machine when necessary without forcing all your users to change their habits or reconfiguring all your client machines.

Finally

Make sure you have monitoring software and systems in place so you can detect failures as soon as (or before!) they happen. I'll cover more about monitoring in a later article.

Next time I plan to discuss system administration for reliability and how you can make your job easier and a little more predictable.

References


ActiveX: An Overview

Recently, while describing the security model of downloaded Java applets and the Virtual Machine implementation in various clients, I was asked if the ActiveX model was similar. I realized at that point that I knew little about the model, except to say “don’t allow it from untrusted sources.” I was motivated to get a better understanding of the technology referred to as ActiveX. This article is a brief of what I have learned and is an attempt to give you a basic understanding of ActiveX technology and how it is used on the Internet.

What ActiveX Is

ActiveX is not an individual tool or set of packages; it is a technology. An extension to Microsoft’s Object Linking and Embedding (OLE) technology, it was originally called OCX (OLE control extension) [1]. It includes a wide range of components: ActiveX Controls, ActiveScripts, ActiveMovies, ActiveVRML, and more to come. ActiveX is often compared to Java. Although there are similarities, the overall methodology is quite different. The actual portion of ActiveX that is similar (but not identical) to the Java content over the Internet (Java Applets) is the ActiveX Control. This article covers the utilization of ActiveX Controls for Internet content delivery.

Comparing Java and ActiveX Controls

Java (and writing applets) is relatively straightforward for Internet delivery purposes. You write some code, compile it to bytecode (a class file), and reference that class file in an HTML document. The client then "hits" the Web page, and the browser automatically [2] downloads the class file. The Virtual Machine implementation in the client will then execute the applet in a "sandbox" [3]. Unlike Java applets, ActiveX Controls do not run in a sandbox after download. The ActiveX Control runs in the user context and has complete access to any resource the user has access to. Remember that
these controls are an evolved form of OLE, thus allowing the same programming functionality across the network as you have on the local machine.

Thus, when you enable ActiveX Controls, you allow full access to the machine and network to which it is attached. Contrast this with Java, which provides only a limited set of functionality for downloaded bytecode. The functionality is tremendous, but the security ramifications are severe. Another obvious (yet needs to be mentioned) point is the development platform used for the different technologies. Java applets are written in the Java language, whereas ActiveX Controls are written in many different languages (Visual Basic and Visual C++ seem to be the most popular). I can foresee an ActiveX Control written in the Java language.

**How You Get the Controls**

ActiveX Controls are embedded in HTML documents with the `<OBJECT>` tag like this:

```html
<html><head> <TITLE>Page with ActiveX Control</TITLE> </HEAD> <BODY>
<object id="filename1.cab", inf]" width=50 height=25
codebase="http://example.microsoft.com/circ3.cab#Version=1,0,0,0"
classid="CLSID:9DBAFCCF-DFE-85CE-00608C297B">
  <param name="_Version" value="12345">
  <param name="_ExtentX" value="2646">
  <param name="_ExtentY" value="1323">
  <param name="_StockProps" value="15">
  <param name="name" value="value">
  ....
  <param name="name" value="value">
  </object> </BODY></html>
```

The attributes in the `<OBJECT>` tag define the specific Control to be used. The extension on the `id` attribute tells us the type of Control we are using. It is a filename or URL; it defaults to OCX if no extension is listed.

OCX is the "portable executable" format, which sends the Control to the client in its executable format. The INF extension is used for interactive installs over the network and allows selective installation of one or more of the desired Controls.

When you hit a page with an INF file, some type of install screen is presented to you (probably something similar to a Microsoft setup screen). The CAB (component download file) enables packaging of multiple Controls in a single download file. They are larger, but compressible, and require real time to unpack and set up on the client. Creating CABs is complex and can cause "perception delay" from a user's point of view while waiting for the unpack and setup to complete.

The advantage of the INF type file is that the downloading of the actual Controls occurs only if selected during the setup routine. The CAB file downloads all the Controls and the INF file as a "packaged" file and then installs only the ones you select. The `<PARAM>` tags determine the Control configuration when it is displayed to the user. The `CODEBASE` attribute specifies the location of the listed control. The `CLASSID` is a unique identifier for the control listed. It describes the version and other information about the control listed in the `id` attribute.

**How It Works**

The client application (currently certain WWW browsers that support ActiveX Controls [4]) uses the WinAPI call CoGetClassObjectFromURL. When the browser sees an
The Download

Once the client completes the download and (if needed) decompresses the file, the client calls the Windows Trust Provider Service function, WinVerifyTrust. This service looks for a “certificate” (pkcs#7 and X.509) inside the Control. The certificate contains the author’s name, public key, and encrypted digest of the Control’s contents. WinVerifyTrust then validates the certificate (in one of two ways), if it exists.

If the certificate is in the “trusted list,” then the Control is accepted with no user intervention. If not, the WinVerifyTrust will traverse the certificate hierarchical tree to the root certificate or first certificate on the “trusted list” that is in that tree. It then verifies the root certificate and the certificate on the Control itself. If the certificate is legitimate and on the “trusted list,” the Control is automatically loaded. Otherwise the user is prompted with a message asking if they want to accept the “untrusted” Control or not. The user chooses whether to install the Control (and optionally add the newly accepted certificate to the “trusted list”). This would enable future Controls signed with that key to be installed with no user intervention. The actual Control files are then installed in a folder named \CCACHE, which is located on the local computer.

Security Issues

There are several security issues with ActiveX Controls. First is the issue of complete resource access by the downloaded Control. Although this is run in the user context, there is still potential for significant damage to be wrought by a malicious Control, especially on Windows95. The other issue is the fact that a major portion of the security is based on the “intent” of the Control’s author/signer. Along with “intent” is the issue of untrusted code. Microsoft touts certificates as the total solution for these issues. The problem with that stance is that it does not scale in a nontechnical user base. A non-technical user would be required to understand the technology well enough to accept code from a specific source as trusted. Some tools are coming out that will allow admins to assist in this effort, but they are still subvertable by the user. The security solutions for ActiveX Controls are not complete by any means, but they are a start. With proper client configuration and user training, you can utilize this technology in a secure manner.

Conclusion

ActiveX is a technology that seems to be here for the long run. It has some very useful features and is a stable platform for development. Although the technology is very appealing, the security issues associated with it are significant. The solutions to these problems, regardless of what Microsoft would have us believe, are not straightforward (this applies to Java as well). With ActiveX, and all new technology, first understand it, then implement it if you deem it a necessity; otherwise wait for it to mature.
Notes:

[1] More information on ActiveX controls can be found in the OLE Control and Control Container Guidelines version 2.0 under the InetSDK/Specs folder of the ActiveX SDK installation.

[2] This is, of course, all dependent on the settings in the browser.

[3] A sandbox is a restricted area in which a program runs. It limits the resources that a given program can access. This is to prevent malicious or inadvertent applet access to system resources that could cause a security problem. The enforcement of this is completely dependent on the vendor’s Virtual Machine implementation, and not the actual Java language itself.


[5] There are different security settings (and thus different promptings), as well. If you have selected no security, then it considers all controls as trusted and automatically installs them.

Spam, Email, Usenet, and Spam

Call it “Unsolicited Commercial Email” (UCE) or “Spam” (gosh but I bet Hormel is unhappy with the new usage of its trademark). By any name, it’s annoying, and it’s taking up more and more of my limited time. I’m not the only one; I have clients taking steps to prevent their servers from being used as email reflectors, clients changing their firewalls to protect their internal mailing lists from it, and friends with personal email accounts who are tired of it. The people who send this stuff glean email addresses from USENET news, Web pages, and mail list servers. They buy and sell addresses among themselves. Because the odds of getting your email addresses off of their bulk lists and keeping them off are near zero (despite their claims “email here to have your name removed”), I’ve got some advice and URLs to help ease your UCE suffering.

First off, USENET UCE: really the only thing you can do is to add the offending news feeders to your site’s aliases; this will prevent those articles from being accepted, because the software assumes that you don’t want to accept articles you’ve already seen. With INN, this involves adding the other hosts to the ME configuration line.

For SMTP, the cream of the current freeware crop seems to be SMTPD, from Obtuse Systems Corporation (as always, see Listing 1). This is normally a part of their Jumper firewall software, but you can download and use it standalone by the terms of the GNU public license or their own license. SMTPD supports a variety of UCE-filtering methods. It allows or blocks mail by:

- domain names that have no DNS A, NS, or MX records
domain names that have DNS served from a given address or network

- regular expression
- IP range

If you have a Trusted Information Systems Firewall Toolkit or Gauntlet-based firewall, here are some enhancements to the smtp program you can apply to help you to rid yourself of UCE. It supports much of SMTPD's functionality.

If you'd rather not have the (perhaps considerable) overhead of a sendmail front end program like SMAPD or smtp, sendmail itself has some settings to help you (if you have a recent version, of course, and if you don't, GET ONE RIGHT NOW because you most certainly have some security-related bug in your version). You'll have to get your hands dirty in the sendmail.cf file, but it's for a noble cause, right?

Lastly, if you aren't the system administrator on a machine being spammed (UCE'd? sometimes the acronym just doesn't fit), your choices are more limited. I use procmail as a mail filter program for other reasons anyway, and it works wonders for this, too.

Here doesn't seem to be much out there to help those who don't use a UNIX host as an email gateway, but check the Web first. I predict a round of upgrades from commercial providers of SMTP gateways and servers real soon now. At least one NT-based filter is available as I write this.

Until this sort of online harassment is made illegal in all countries (and while I'm dreaming, I'd like world peace and Bill Gates's fortune, please), software like this will be our best defense against those bloody vikings shouting "SPAM!"

Listing 1: URLs To Visit
ntsmap documents and lots of links:
<http://www.vix.com/spam/>
<http://www.caucer.org/>

Sendmail and sendmail antispam:
<http://www.sendmail.org>
<http://www.sendmail.org/antispam.html>

Map enhancement page:
<http://www.cih.com/smap-hacks/>

MTPD:

F T-based filters:
<http://www.sica.com/freestuf/mfilter.htm>

Procmail-based filters:
<http://spitfire.ecsel.psu.edu/~gsutter/junkmail/>
<http://www.best.com/~ariel/nospam/>
I read the responses to Andrew Hume's "Whither Now, USENIX?" column in the August 1997 :login: with interest. While I had little problem with Andrew's original column, some of the responses disturbed me. It appears that many people feel that USENIX should only deal with UNIX. Let me tell you a story about why I think this is not true. I like telling stories.

Back in the late '70s, I decided to go back to school and learn about operating system design (a course that didn't exist when I graduated). The textbook was about the parts of IBM's operating system that were not proprietary. We learned how to program a channel controller, for example. Meanwhile, we struggled with building a multitasking operating system for a PDP 11/45 in lab.

You can't imagine how much this bothered me. I had come into possession of a Zilog Z80 CPU manual and realized that this little Intel 8080 clone represented the tip of an iceberg. No longer would IBM dominate the world of computers -- we were already building entire CPUs on a single chip! While learning about channel controllers was not totally irrelevant, it was certainly not the wave of the future.

I approached the course's graduate assistant earnestly and asked if there wasn't a better example operating system that we could be studying. He said there wasn't and, besides, IBM and its operating system were what the professor had planned for the course. I quit the course and got a job working with a company using the Z80 CPU in embedded systems. I got to write parts of a multitasking operating system, and I got paid for it too.

It wasn't until I moved to California in 1979 that I learned about UNIX. Several years later I got back a copy of a chapter of my book about UNIX system administration which had been reviewed by the very professor who had been teaching IBM 360/OS in 1978. He said that my book was unnecessary, people could just read the manuals. Keep in mind that in these days, UNIX documentation was in three volumes, and totaled perhaps 1,000 pages.

The Point

In 1978, I really needed -- in fact, a lot of people really needed -- UNIX. The UNIX operating system provided an accessible, portable design for operating systems that could be used on microprocessors. By learning in my class how the UNIX operating system worked, we would have learned more about the design of operating systems. Instead, we were pecking around the edges of the proprietary, secret, IBM operating system.

The USENIX Association was founded to help people work with UNIX. Those manuals I mentioned (mainly just man pages in the beginning) came on nine-track tape, and if you didn't have a CAT typesetter you couldn't print them out. And the UNIX operating system itself was unstable. System crashes were common, and fsck didn't even exist yet. (By the way, this should help you understand why the sync account exists -- as a last ditch method for preserving file system consistency as the system crashed. If the system manifested evidence of instability, you entered sync, RETURN, at any terminal.)

Through USENIX, people could gather together and share code, documents, and experiences. These people recognized the value of UNIX as a tool that helped them to use the minicomputers and microcomputers of the early '80s and learn more about operating system design. Source to the UNIX system could easily (and rather cheaply if you were a university) be licensed from AT&T. Early BSD implementations (still requiring
the AT&T license) cost $50 to cover the cost of the nine-track tape, handling, and shipping.

Does this mean that USENIX should only focus on UNIX? I don’t think so. UNIX was the early focus, but the real purpose around this focus wasn’t UNIX as religion, but advancing the state of the art of computing. UNIX was the best tool there was in the late ’70s, the ’80s, and the early ’90s. But it is not the only solution. UNIX is, and has been, only part of the big picture.

USENIX conferences have always included papers about compilers and compiler techniques. Operating systems that shared the UNIX API (application programming interface), such as MACH or Spring, have been included. New high-level languages, networking protocols, file systems, and operating system designs have all become an important part of USENIX conferences. As more people participated, other, more narrowly focused conferences were organized. The NT conference in Seattle is just another manifestation of the direction USENIX has taken over time.

All of the hype surrounding Microsoft’s NT operating system can easily set me off. From the letters received in response to Andrew’s column, this is true for many other people as well. But I want to get beyond religion. NT was intended to be a replacement operating system to DOS. I don’t think that any of us will claim that DOS was wonderful or that the Windows API which sat on top of DOS broke new ground either. Together, Windows and DOS provided a standard base for mass market applications. And now Microsoft is stuck with it.

NT itself represents the New Technology the letters purportedly stand for. NT is modular and message-passing (but not quite a microkernel), and designed with portability in mind. The device driver interface is supposed to make writing device drivers simpler (I don’t know about that). And the system is much more reliable and robust than the one it replaces. In other words, there are valid examples to be found in the implementation of NT, and future operating system designers had better take note of what it does right, and what it doesn’t.

I personally think that NT is overkill for desktop applications and that something else will replace it. Keep in mind that this is one of the reasons that UNIX did not succeed in the desktop marketplace either. At the same time, NT’s reliance on the Windows API makes it less suitable for server applications, while at the same time reducing security. (Windows applications install programs and files anywhere, making trojan-horsing NT systems a snap.) This is the legacy of Windows that will be Microsoft’s downfall, even as it is its most important asset today.

**Whether USENIX**

The USENIX Association has often been at the forefront of research into computing systems. On a more practical note, USENIX members, through papers and tutorials, have provided workable solutions to many system administration and programming problems. The key to these solutions has not always been UNIX, but open, nonproprietary solutions. Note that the key to the Internet’s success has also been its openness and nonproprietary design.

Linux and the BSD versions today provide the training tool I was looking for back in 1978. Whether Linux is ready for business use is a moot point; it is a great learning tool, and businesses are already using it. One reason is that Linux is cheap; free from the Internet or about $40 for a CD. Another is that the Linux es2 file system just screams – because it cuts corners which the more conservative designers of the fast file
If we just decide that the only thing we will talk about or work with is UNIX, we have just made ourselves obsolete.

system, a.k.a. the UNIX file system, didn’t take. Does this make it wrong? If es2 can be proven to be robust and reliable, as well as being really fast, it would be stupid to ignore it just because it is not UNIX. And even if es2 turns out not to be a robust solution, the reasons behind the failure would be an important lesson in file system design for future programmers.

The story of computing won’t end with UNIX. If we just decide that the only thing we will talk about or work with is UNIX, we have just made ourselves obsolete. The world moves on, and the world of computing moves faster than most. I don’t want to get left behind. I have enjoyed the ride, the neat stuff I have learned, the cool toys I have played with (or now own), and the riches I have collected by understanding the technology.

While it might be comforting to stick with what we already know, we will be usurped by a younger generation of people not stuck on old ideas and beliefs if we do. I for one am not ready to say, “It’s already over, we have discovered everything interesting and/or useful.” I want to keep an open mind and stretch myself in new directions. I know that this will pay off in the end even if it will be uncomfortable during the process.

So whither USENIX? To the frontiers of advanced computing. And I plan to be along for the ride.

A Note on Java
As Terry Slattery has taken over this issue’s “Using Java” column, I will use this soapbox to talk about some “recent” events. Microsoft has decided not to distribute the Java Foundation Libraries (JFC) with its operating systems as the libraries represent “another operating system.” Nice going, Microsoft.

The JFC are required for true portability of Java applications. Microsoft will still support Java, but only with their own Windows-based Java libraries. The JFC is “bloated,” says Microsoft. Yep, about 1.5 megabytes of bloat which would hardly be noticed on a typical Windows or NT system, especially if some applications have been installed.

But Microsoft is right about the JFC being another operating system, at least in a way. The application programming interface (API) represents the operating system, and Java with its JFC truly represents a different operating system, one not limited to the Wintel platform. I am only surprised it took Microsoft so long to decide to do this.

Of course, Netscape Navigator will support the JFC. Perhaps Microsoft’s refusal to include the JFC will make Navigator an essential part of Windows desktops, instead of Internet Explorer.

In tangentially related news, Bill Gates made a virtual appearance at MacWorld in August (to numerous boos) and agreed to support the Mac platform by updating its applications for the Mac. Now that Windows has finally approached the functionality of the Mac user interface (seven releases of Windows later), Microsoft has decided to help keep the Mac from dying. Some cynics remarked that Microsoft is only doing this to avoid anti-trust litigation, Apple being the only real competitor to Windows on the desktop.

Having just watched Robert Cringley’s “Triumph of the Nerds,” I better understand Mr. Gates. Bill Gates might or might not be a nice guy, but he is a real competitor. In other words, he is not saving Apple out of the kindness of his heart. It will be interesting to see how this plays out.
discussing security

An Interview with Char Sample and Mark Teicher

Rob What do you consider a secure environment?

Mark This could be a whole article in itself. Different environments require different definitions of a so-called “secure environment.” Secure environments for an engineering firm are different from a military/government type of establishment. Engineering firms may require employees to wear name tags, and military/government establishments may require you to wear a name badge, including a picture ID, fingerprint, and color coding. This is just the tip of the iceberg for my definition of a secure environment or my consideration of a secure environment.

Everyone has an idea of what a secure environment should be, but that is based on what those people have observed in their work and what they hear, what they see, and what the industry is leading us to believe. In the business world, companies and industries define their own ideas of what is a secure environment. In defining a secure environment, one must consider the people, the hardware/software, the physical plan, and the underlying mission or business model of the company in order to clearly define what the company/industry requires in terms of a secure environment.

Most of my work over the last few years has been examining physical security. Physical security, in my mind, is where most of the inadequacies start. If a company wants to secure what is inside, but would like it to be convenient for employees to arrive and leave as they please, then compromises must be made. Regrettably, too many companies do not think about security until something traumatic happens (i.e., employee goes postal, a very high level person is let go, or someone who knows a lot about the company leaves). Security should be on the forefront of a company’s mind, not just an afterthought.

There are risks at each level in trying to secure an environment. It ends up being about what you are trying to protect, from whom you want to protect it, what information you are trying to protect, and how much are you willing to invest to protect that information.

The one environment that I characterize as meeting my level of security requirements is probably Fort Knox, a high-level military installation protecting the gold supply of the United States. The vault room that holds most of the gold allows only a few select people in and out. Clearances must be sought and justified, security officers are informed, ID cards are marked, and on and on. Think about what they are trying to protect, how much they spend to protect it, and how they protect it. If people thought about what the military/government does to protect its gold supply, they might have a better idea of the scope of the security problem.

Char Wow! Talk about a broad question!

A secure environment is really a difficult thing to define. Not only that, but its definition varies based on security needs. An argument can be made that there is no such thing as a really secure environment. Instead, there are certain levels of risk, which we are all content to live with, within our own environments.

Of course, each person is unique, and each person has his or her own definition of a secure environment. Translating all of these definitions into a coherent policy to form one secure environment is bound to leave some people less than thrilled.

Editor’s note: This interview was conducted electronically over the summer among Rob Kolstad, Char Sample, and Mark Teicher.

Char Sample
<Char_Sample@notas.pw.com> is Manager, Price Waterhouse LLP, Enterprise Security Solutions.

Mark Teicher
<mlit@clark.net> is an Internet and firewall security solutions consultant with over eight years of experience.
For example, let’s look at Company A, a large multinational corporation that decides to launch an Internet initiative. Because many different departments are affected by this initiative, meetings are held and everyone provides input. In these meetings, various interests and concerns are voiced. These interests and concerns may range the entire spectrum of “nothing connects” to the other extreme of “everything connects.” The final decision usually falls in the middle of the extremes; therefore, neither side is 100% happy or 100% disappointed. However, both sides must change their present policies to accommodate the new policy.

Enclaves within our environment can be made secure, but they, too, have to interface with the less secure. This is where issues of multiple levels of trust usually come into play. How do we deal with multilevels of trust and moving between these levels? In short, which path do we follow: risk avoidance or risk management?

The short answer is both. We don’t take foolish risks (risk avoidance), but we may allow for some things we are not 100% comfortable with in order to simplify our lives.

For example, I’ll use my house (because it’s easy to understand and it drives my husband nuts whenever I use the house example). My house may be in most cases a protected environment. I have locks on the doors and windows, a canine alarm system, and the construction is pretty decent. The house has withstood some strong storms and some rather significant snowfalls. However, should a twister come by, maybe only certain sections (protected enclaves) of my house would end up intact.

The same concept applies to the business world. An entire entity is practically impossible to protect, especially in the case of large corporations. However, one can still have protected enclaves within the corporation.

Let’s go back to Company A. Perhaps the payroll group is requesting absolutely no connectivity while R&D is requesting full open connectivity. Furthermore, suppose the compromise leads us to authenticated connectivity for specified services. Here we have a policy that is in the middle of the two extremes. The group that wants no connectivity may now want to purchase additional protections. The group that wanted full open connectivity may now find itself trying to “go around” the security controls. In a large business, this behavior is very difficult to control.

In the business world, the environment encompasses management, employees, the physical plant, all assets, and the interaction among these items (which would also include business processes). The “enclaves” are defined within the processes. Within the enclaves there exist management, employees, physical objects, logical objects, and interaction among these items. By examining not only objects but also the interaction between those objects, we are able to provide a systems approach to dealing with the issue.

A systems approach, when performed properly, should take into account all needs, and address not only all of the needs but also any interactions between objects. The whole picture must be examined and addressed. Addressing the security of these items to the acceptable level of risk is dependent upon the policies (defined in processes). Here is where some places use security matrices to assist in this process.

A security matrix requires the site to list all assets in one column and all of the risks associated with those items in the top row. Controls for each risk are numbered and defined outside of the matrix. Within the matrix, the controls associated with the risk are matched with the item and placed in the box.
Security matrices are a good tool to assist in this; however, there is a strong need for up-front planning in all aspects of security. The security matrix requires the planners to be aware of all of the potential risks. If the planners are unaware of a particular risk, then it will not make it onto the matrix; therefore, there is no way of being able to control the aforementioned risk. Additionally, these matrices require periodic reviewing and updating as new risks emerge. This is particularly true in the case of software, but certainly not limited to that area.

Good security planning is part of a design process. Unfortunately, this is an area that could use some improvement. TQM and some other quality programs were supposed to fix this, but I have yet to witness it. Quality programs concern themselves with the processes in place and rely on those to achieve quality. This is an inductive approach. Good security would demand both an inductive and deductive approach.

Unfortunately, my experience has shown me that, when faced with the cost of doing security right, many places balk and do a cost/benefit analysis without thoroughly addressing the risk analysis. The outcome then becomes one where security is compromised from the start. When this happens, you can be sure that political and financial concerns will win over security concerns. Of course, when an incident occurs, the business is willing to throw all sorts of money at the problem to make it go away. I call this the “stupidity tax.”

A truly secure environment employs checks and balances for all processes and transactions in addition to using processes and procedures to minimize impact when breaches occur. That said, actually implementing a truly totally secure environment across a large organization is at best a difficult and complex task that is never static and darned near impossible to enforce.

Rob How do you go about evaluating all of the items listed above?

Mark Most organizations try to establish the ability of defining a single enterprise-wide policy that integrates all aspects of network and physical security. Sometimes this is a very hard scope in itself to define.

This type of evaluation takes on many challenges, taking into account the nature of the business model of that particular client, their working environment, communications, and so on. Some of the initial steps include defining access control, authentication, connection control, auditing, enterprise-wide management, and communication between departments.

Access control decides what people have access to what rooms and what devices and the level of access they have. Authentication means verifying that the user does indeed have access to hardware/software and the particular room and verifying that the user is who he or she claims to be (e.g., “show me your ID”).

Connection control defines who has connections to what machines or what rooms within in particular environment. Auditing defines the level of accountability or alerts that one environment may have in place to detect an intruder or what people are actually doing within a particular environment. Enterprise-wide management defines who has what power to decide on a particular network/security policy and has the power to enforce it when things go awry. Communication between departments defines the channels of talkability or conversation across departments to ensure that one depart-
ment is not doing something another department may be doing. All of the above listed items play into the role of evaluating a secure environment.

**Char**  First, begin by asking, “What is it I am trying to accomplish here?” Otherwise, project scope will kill you. This first step is an iterative process when defining enclaves, components, and interactions within the environment.

Next, I’d ask how would I plan on getting from here to there based first on needs and then possibly address desires. “What tools (products) are available to assist in this journey?” Here is where a security consultant needs to look at the tools and ask, “How do these tools really work” (and not just reading the glossies or a matrix either)?

For example, let’s take a look at firewalls. Datacomm did their throughput study, but they were not consistent in the comparisons. Throughput can be affected by the following: hardware capacity, speed of the line, operating system, and the efficiency of the actual software being tested. However, not all firewalls support the same platforms. Yet, when the testing was completed, Datacomm placed the results in a nice and pretty matrix. Information was reduced to a sound bite.

Once the knowledge of how the tools work is in place, you will move on to the question of why was the tool designed to work this way—which in turn should lead to a bunch of “what ifs.”

For example, let’s revisit our firewalls. Various firewall types were initially designed to address a set of security needs. As the industry changed, the firewalls began pulling on their designs without re-examining the basic foundations. Basic packet filtering firewalls became “stateful,” and proxy firewalls began employing packet-filtering capabilities. A matrix will show that the major firewall vendors all offer proxying and packet filtering. However, if the site wishes to use packet filtering, the consultant’s job would require that they provide a packet-filtering solution, not a proxy solution, which can be modified.

Firewalls are moving toward a middle ground of becoming hybrid, but I have not yet seen a “true” hybrid firewall emerge in the market. The funny thing is that a hybrid firewall makes a lot of sense because most sites security needs fall somewhere in the middle of the two extremes.

**Rob**  What type of solutions do you provide or experience do you bring with you?

**Mark**  Internet security and security itself comprise ever-evolving fields. Some new company is always claiming, “We can now protect your site better than our competitors” or “Buy our product and be secure on the Internet.” If you really want to be secure, refer to Marcus Ranum’s ultimate firewall (a pair of wirecutters).

In actuality, companies want Internet access and a secure environment. How do I give them that warm and fuzzy feeling? I talk with the customers and ask them about their business model: how their business is defined, the projections of growth in certain areas, the projections in network exposure, capability, growth, and redundancy. All these items factor in to a best fit, best product(s), best architecture, and best solutions concept. There are many different solutions and many different approaches in developing solutions.

Evaluating every approach or solution can be costly and very time-consuming, but I like to provide unique and feasible solutions and not vaporware. Usually about once a
month, I evaluate some new vendors’ products or check some of the various security mailing lists so I can give the customer many solutions but not very complicated ones. Steve Bellovin and Bill Cheswick emphasize that one should never develop a security plan or policy that is very complicated. Try to keep things simple and logical, not overly complex. Having spent some time on both the development and users side of the house implementing what I helped develop, I found it quite enlightening that sometimes the integrity and coding of a firewall/Internet product can be scrutinized by a customer's vision of what a firewall/Internet product should do and shouldn't do. I attempt to alert them about the security implications of all the known variables and let them decide. Sometimes it works; sometimes it doesn't.

Char  This is tough. It almost sounds like a chance to plug my employer's service line. Seriously, the experience of having installed 150 firewalls (most of them Gauntlet) and having to do some strange things in order to make the firewall fit the site's needs, along with the experience of coding, gives me a slightly different perspective.

Having spent time in the coding environment at large defense contractors, I know a few things about how code gets written. Too often, it is not a pretty process. I have also worked in the commercial world coding, and some of the practices there are even worse because there is no time for official reviews and walkthroughs. Coders will do whatever it takes to make something work; this generally does not translate into secure coding practices.

The same can be said for installers. Often when onsite, the customers may not have a complete understanding of how the firewall was designed to work. Once the firewall has been installed, the customers see firsthand how the security policy is being enforced. This sometimes conflicts with their perception of how the firewall should work, so they want the firewall configuration modified.

The best that I personally can supply is a working brain. By that, I mean most people search for criteria on something to understand how it is supposed to work. I balance that against the needs and provide a solution. When consulting, your first duty is to the customer (obviously), and the customer's first duty is to the business. Therefore, a good consultant will first try to understand the business, next the soft issues (personalities, political agendas, etc.) along with how they interact before making any recommendations.

Matching criteria without asking "how," "why," and "what if" is not a great model in the security world. Breaches in security occur when users (both invited and uninvited) start asking these same questions.

I am a strong proponent of good planning. Good, sound planning heads off many of these problems before they have a chance to happen. Additionally, good planning requires a good review cycle. The best plans need to be periodically updated.

Rob  What problems have you found?

Mark  The problems I encounter most frequently are political, management intervention, management cluefulness, and technical expertise. I think Char is going to tackle the political piece of this question. I will tackle the management intervention, management cluefulness, and technical expertise of this question.
Some of the installs and solutions I have provided over the years usually get blocked by management intervention. Management sometimes tries to step in during the installation or implementation of a firewall and Internet solution in order to keep their fingers in the pie. Management usually does not have the knowhow or care about where the solution is going to be installed, but usually cares about, "How is this going to affect my management team or budget?"

Recently, I installed a firewall and Internet solution where this did happen. Firewall and Internet solution were installed. We were installing a logging machine inside the firewall perimeter but ran into problems with management intervention of the "that should not go here or it will affect business" variety. Our logging machine was never installed, and we could no longer keep track of the traffic or audit trail of the firewall properly due to the incident.

This also brought out the notion of "management cluefulness" because management did not know enough of the technology to understand that auditing/logging is important for many reasons. Again, this also displays the example of how management simply did not have the expertise to fully understand how the installation/implementation would affect their life, but expected it to be a very simple affair, just like plugging in a kitchen appliance.

**Char**  Many times, the problems I have encountered have been political. The person who is responsible for securing the environment knows all too well of the risks associated with the environment. However, if that person cannot convey this knowledge to management in the language that management understands, the security person's knowledge is worthless to the company. The company needs to hear about risks and costs associated with the risks. Simply explaining in detail a risk to a manager is not sufficient; he or she needs to hear the cost (financial and legal) associated with the particular risk along with the cost to mitigate that risk.

Another problem I have encountered is that in most of the places I have installed firewalls, the buying decision is usually not made by the person who works with the equipment. The buying decision comes from management and purchasing. If decision makers are not very technical, they can be taken in by aggressive marketing and end up forcing a product on the administrator.

When I was first installing firewalls, I used to say the criteria for a successful install is if the "big guy" can surf and read his email, he doesn't care about anything else. This is not entirely true. These are simply the yardsticks that make people feel good. Most managers are not going to run a port scan to see if the firewall is in. They want to know that business can resume.

The criteria for a successful firewall install unfortunately have nothing to do with security and everything to do with keeping a business functioning. The management does not want to hear that certain holes had to be punched into a firewall to make some of the other services work -- that is until they buy some penetration services. Then they want to make sure they are 100% secure in the controlled environment against ISS, SATAN, or Ballista. In this case, they are testing to see if that $20,000 spent on securing the site will withstand a scan that the box was programmed to protect against anyway.

I find this thinking a bit shortsighted. If I had purchased a firewall for my site and I wanted to validate it, I would take a systems approach to evaluating it. This would involve inductive reasoning (attestation, validation, and verification) and deductive rea-
soning (scanning tools). To do one without the other lacks a certain completeness. I would want to examine the sources for the security code (under NDA, of course) and determine whether the code is security proactive or security reactive.

Security proactive code would be cleanly and simply coded in compliance with good coding practices. Variables would be checked, and there would be handlers for unexpected conditions. This code cannot only withstand known attacks, but would also be positioned to withstand many unknown attacks. (A good example of security proactive code would be smap/smapd. By allowing only certain commands, smap/smapd started with a proactive premise. A user who tried to issue a command that was not supported simply received an error response. Smap/smapd is now over three years old and has had some modifications, but the basic idea still works. During those three years, there have been problems found with sendmail, and many of them had no effect on smap/smapd. Now that is security proactive code!)

On the other hand, security reactive code would be able to withstand known attacks. However, this code may not be in compliance with good coding practices. Unexpected conditions may result in a security problem (breach). This code would not be positioned to withstand unknown attacks. It may withstand them and it may not, a prediction cannot be made. (I will not give examples of security reactive code here because that could open up a whole can of worms; however, I think the readers can figure out what I am referring to by simply reversing the criteria in the prior paragraph.)

Finally, all of this needs to be done within the context of the business. I don’t run a business; therefore, I don’t have to worry about the bottom line. If I did, maybe I’d be a little more willing to go with some quick fixes.
Surveys Online!

I’d like to show a very simple solution that you can expand to create a pretty cool new area on your Web site: an online survey system.

The basic idea is simple: a Web page includes a survey question that’s really a form, and the answer is added to a log file that contains the combined answers from all survey takers. To display the survey results, you simply open up the log file, tally all the yes and no answers, and display them on the screen.

The HTML Frontend

The core of the HTML that you need to ask the survey question is shown here:

```
<html>
  <body bgcolor="#FFFFFF">
  <center>
    <font size="6"><b>Take Our Simple Survey!</b></font>
  </center>
  
  <p>
    <b>I think that watching television improves your mind:</b>
  </p>

  <form method="get" action="take-survey.cgi">
    <input type="radio" name="answer" checked value="T"> Yes
    <br>
    <input type="radio" name="answer" value="F"> No
  </form>

  </body>
  </html>
```

Note that if you strip out all the formatting, the entire form boils down to a five line form:

```
<form method="get" action="take-survey.cgi">
  <input type="radio" name="answer" checked value="T"> Yes
  <input type="radio" name="answer" value="F"> No
  <input type="submit" value="tally my answer">
</form>
```

This appears on screen as shown in figure 1.

This is the simplest of all possible surveys, of course; a single question and two possible answers, true or false. (Well, in this case it’s yes and no, but the concept is the same!)

Nonetheless, it’s an easy little device you can use to add some interactivity to an existing page. Sites like Parent Soup [http://www.parentsoup.com/] use it to very good effect.

Receiving the Survey Answer

To receive the survey, the resultant CGI program – in C as usual – is straightforward because we’re insisting on using the METHOD=GET, which ensures that the answer is passed in the QUERY_STRING variable, and because we don’t have to parse the resultant value.

When the user clicks on the “tally my answer” button, the URL that is built looks like:
```
take-survey.cgi?answer=T
```
which means that the QUERY_STRING variable looks like:
```
answer=T
```
so it's a simple step of getting the QUERY_STRING variable then looking at the eighth letter to see if it's a T or F, as you can see in the code snippet below:

```
main()
{
    char answer, *cp;

    if (!cp = getenv("QUERY_STRING")) == NULL)
        error("no query string?");

    answer = (*((cp + 7) == 'T'); /* test the first character only */
    add_answer(answer);

    show_results();
}
```

In this case, as with many other CGI programs I develop, I've opted to have an error() routine that displays to the user the specified message and then immediately exits from the script. Makes handling errors quite a bit easier.

With this code, the only way that you can see the answers to the survey are to answer it yourself, which isn't necessarily the best way to code this. An easy addition here would be to indicate that a zero-length QUERY_STRING simply shows the results instead of the error message, which you'd accomplish by changing the above code to the following:

```
main()
{
    char answer, *cp;

    if (!(cp = getenv("QUERY_STRING")) == NULL) ||
        strlen(cp) == 0)
        show_results();

    answer = (*((cp + 7) == 'T'); /* test the first character only */
    add_answer(answer);
```
show_results();

I am taking advantage of one trick here: on a conditional statement that has a logical OR between its expressions (a || b) the second expression isn’t evaluated if the first is true. This means that we don’t have to worry about the situation where the getenv() returns a NULL that we then feed to the strlen() function. If you have any weirdness with this CGI script, this is a likely spot for the problem!

Adding the Answer to the Log File

Once you have the answer as either true or false (1 or 0, of course), you need to open and append it to the log file. This is done with:

```c
add_answer(answer)
char answer;
{
  /** append the user answer to the survey **/

  FILE *fd;
  if ((fd = fopen(LOGFILE, "a")) == NULL)
    error("couldn’t open log file");

  fprintf(answer ? ‘T’ : ‘F’, fd);

  fclose(fd);
}
```

I could directly write the value of answer in the log file, as a 0 or 1, but it would take up more space (each would be an int) and it’d be impossible to edit or view with regular UNIX utilities. Instead, the inline conditional answer? ‘T’ : ‘F’ evaluates to the ‘T’ character if the answer is 1, and ‘F’ if answer is 0.

Because of how most servers have the permissions set, you’ll probably need to create the log file with the touch command and make sure it has permission so that your CGI scripts (which will probably run as user Web, WWW, or similar, rather than you) have permission to append. I use chmod 777 *.log to solve the problem, making sure that the directory is mode 755 to avoid too much trouble.

Showing the Results

Once you have a growing log file, you unquestionably want to show survey takers the answers compiled so far. That’s done with the show_results() routine:

```c
show_results()
{
  FILE *fd;
  int answerstring[2];
  char ch;

  answerstring[0] = answerstring[1] = 0;

  if ((fd = fopen(LOGFILE, "r")) == NULL)
    error("no log file to show results?");

  while ((ch = fgetc(fd)) != EOF)
    answerstring[(ch == ‘T’)] ++;

  close(fd);
  printf("Content-type: text/html\n\n");
```
printf("<BODY BGCOLOR=#FFFFFF>n*");
printf("<CENTER><font size=6><b>Survey
    Answers</B></font></CENTER>n");

printf("Answers so far:<p>n");
printf("%d answered TRUE<br>n", answerstring[1]);
printf("%d answered FALSE<br>n", answerstring[0]);

printf("</BODY></HTML>n");

exit(0);
}

Here you can see that the file is read and tallied each time, with the true and false
answers stored in the answerstring array. Also note that the usual "Content-type:
text/html" must be output before any of the HTML is produced, as always.

The final routine to see here is the error function:

error (string)
char *string;
{
    /** show the error in a reasonably cool format **/

    printf("Content-type: text/html\n\n");

    printf("<BODY BGCOLOR=#FFFFFF>n*");
    printf("<CENTER>n");
    printf("<h2>Error Encountered:<h2>n");
    printf("<h3>%s</h3>n", string);
    printf("</CENTER>n");
    printf("</BODY></HTML>n");

    exit(0);
}

Figure 2 shows the answers to the survey after a few people have seen and responded

![Survey Answers](image)

**Figure 2: Bare-bones survey answers**
"The WebMaster" can be found (together with all the previous columns) online – including the actual survey – by visiting the Intuitive Systems Intro to CGI Programming area at:
Interested in a specific CGI programming topic? Drop me a note and I'll try to work it into an upcoming issue of login.

For the Future
This is a very primitive solution, but you can see the skeleton of how to create a quite sophisticated online survey system, one that can display the results as percentages, bar graphs, you could even have a pie chart if you were motivated!

The biggest problem with this, however, isn't that the results are displayed as boring text – which isn't too horrible – but rather that the results don't reiterate the original question. This is a major drawback of this solution and one that can only be solved by either having the take-survey CGI program sneak into the HTML source of the page and dig out the question (which would be tricky) or to have the question itself – and perhaps its possible answers – stored in a separate file.

The latter solution sounds good, except: if you want to have the survey on a static Web page, perhaps your home page, then you end up with the survey question in two places – that page and the info file – with all the usual challenges for ensuring they're consistent.

If the survey page is dynamic, of course, it can very easily read the same data file that is used to display a nice answer, which solves a lot of the problem. In fact, you can see this exact strategy (albeit with a slightly different application) at work online at <http://www.intuitive.com/origins/> where a shared data file contains both the question, possible answers, and the correct answer for each of the many questions in the game.
using java

TTCP (Test TCP) is a network-testing program that I've been involved with since the early 1980s. The UNIX version of TTCP started out as a TCP throughput-testing program. It makes a TCP connection, pumps data through the connection, then reports statistics on the throughput. Many individuals in the UNIX and networking communities have enhanced it over the years. Today, the UNIX version handles both UDP and TCP tests and includes a lot of statistics reporting on various components of the transfer.

I wanted to learn Java and so decided that the combination of features required to implement TTCP would be a good exercise. A graphic interface would be useful for window-based systems, threads would be needed to handle the user interface in parallel with the data transfer component, and the networking component is absolutely required. To top it off, I decided to do the development with one of the integrated development environments to learn how they work.

The resulting Java version has both a GUI for use on window systems and a command line interface for use on remote systems where you may not have a windowing system available. It currently handles only TCP connections and, unlike the UNIX version, cannot handle data piped through it. Both the UNIX source and the Java executable are available from the Chesapeake Web site, <http://www.ccci.com>. The Java source can be found at <http://www.usenix.org/publications/java/javaindex.html> on the USENIX Web server.

Development Environments

I come from a UNIX background where my favorite development environment is emacs. However, our company issues a PC to everyone who travels (as I do). So one of the things I wanted to learn was whether the Java Integrated Development Environments were usable. I selected the Symantec Cafe development product and JDK 1.0.2 version of their compiler.

One thing I've found about PCs is the lack of real documentation. It used to be that UNIX always had the bad reputation on documentation, but these days, when you purchase a PC product, you often have documentation that gives you 25-50% of the knowledge you need. For Cafe, I had to visit the Symantec Web site and download a tutorial document that walked me through the use of the product.

Once I had the documentation, I was able to be productive with their development environment in about three hours. During this time, I made three attempts at building the TTCP graphical interface I wanted. By the third version, I knew enough about how the system worked that I was able to build the GUI (resulting in three pages of generated code). I consider this to be a very short learning curve! In the future, I expect to be able to create prototype GUI interfaces very quickly.

The other advantage to the Cafe development system is that it compiled the code in under five seconds on a desktop Dell PC with 24MB of RAM and a 90Mhz Pentium processor. Compiling the same code on a Sparc20 running Solaris and JDK 1.1.1 took tens of seconds. The development environments tend to preload the entire environment, and many provide incremental compilation, which helps increase their speeds. Even with this startup load, I didn't find that starting the application took much time.

One downside to a development environment is that you will have to follow the environment's rules for modifying the resulting code. I made the mistake of editing some sections of the code that the development environment was using and found that Cafe...
refused to load the result. Another “gotcha” is that development environments do not always use the best features of the underlying programming language. For example, the Cafe version I used positions the graphical items using explicit locations. This prevents TTCP from appropriately handling resizing events that would be better handled by one of the Java AWT layout managers.

This isn’t enough of a detriment to prevent me from using a development environment for rapid prototyping and is likely to be fixed in future versions. To give you an idea of the development time, I started with a GUI design on paper and was able to produce the GUI and resulting Java code (see below) in under 20 minutes. Development environments, although they have some limitations, are very useful for rapid prototyping and “proof of concept” applications. They don’t replace good software design techniques – I had already designed the GUI layout and knew exactly what I wanted when I started.

**Threads**

Native support of threads within Java is nice! There are even some good books available about the topic (e.g., *Concurrent Java Programming* by Doug Lea is a good text). The JDK documentation doesn’t go into enough depth on the subject, so I wound up experimenting to determine the operation of some things. For example, in an object-oriented world, one would like to construct a thread object, use accessor methods to set a number of variables, then run the thread whenever it is needed. I found that once a thread stops, it cannot be restarted. So I created a separate class to hold all the variables needed by a thread and had the thread access them when it started.

It also took me a while to discover how to implement interprocess communication between the GUI and the worker thread (which is named TransThread). Chesapeake TTCP operates in the following manner: The GUI runs and allows the user to modify the startup parameters. Then, when the START button is pressed, a new TransThread is started. Simultaneously with starting the new thread, the GUI modifies the appearance of the buttons so that only the STOP and QUIT buttons are enabled, giving a visual indication that a test is running. When the STOP button is pressed, the GUI performs a stop operation on TransThread within the following code fragment:

```java
public void clickedButtonStop() {
    System.err.println("Stop");
    if (trans != null) {
        if (trans.isAlive()) {
            trans.stop();
            trans = null;
        }
    }
    ButtonState(Params.STOP);
    Status.appendText("Transfer interrupted.\n");
}
```
This stops TransThread, resets all button states to the START state, and displays a status message in the Results section of the GUI. But what happens when TransThread reaches the end of the transfer? It needs to inform the GUI of its completion so the GUI can reset the button states.

The method that I came up with, which I later found described by Doug Lea in Concurrent Java Programming, is to have TransThread accept the GUI object as a parameter. When TransThread finishes a transfer, it calls the transferComplete() method within the GUI to reset the button appearance.

```java
TransThread(ParamsClass p, ttcpControls c){
    Controls = c;
    Params = p;

    ... private void savestats(int i) {
            Params.setBufCount(i);
            Params.setStart(start);
            Params.setEnd(end);
            if (Controls != null) {
                Controls.transferComplete();
            }
        }
    ...
}
```

The method within the GUI that resets the button states looks like this:

```java
public void transferComplete() {
    ButtonState(Params.STOP);
    Status.appendText(Params.getStats());
}
```

### Networking

Once the GUI was working, it was time to add the networking code. There are several good examples available on the net, and I used them to help point me in the right direction. The Java networking model is certainly easier than the C model, but what you give up in ease of use also sacrifices in low-level access. For example, Java currently doesn’t have access to ICMP, so it is not possible to build a true Java-based Ping or a Java-based Traceroute (it is possible to build a native method to access ICMP, but I discount this approach because it isn’t portable).

Creating the socket is greatly simplified by using the Java Socket class, which includes the ServerSocket class for servers (receivers) and the Socket class for clients (transmitters). Once the socket exists, I created a DataInputStream or DataOutputStream from the socket to provide a high-level read/write stream interface.

There are a number of read/write methods available for these streams, but I used the DataOutputStream.write() method to write data on the sender and DataInputStream.readFully() to read data on the receiver. The readFully() method blocks until it has read the requested amount of data. To make the application robust, you have to include a lot of exception catching and handling. It’s not efficient to try/catch each individual Java exception, so I include a whole block of statements within one try block and use several catch clauses to handle the important exceptions. The finally clause cleans up and saves the statistics of the transfer. The receiver code is shown below. The transmitter code is nearly the same:

```java
if (TransmitReceive == Params.RECEIVE) {
    try {
        if (Server == null) { // Create receive server socket
            ...
```
Server = new ServerSocket(Port);
}
report("Receive: buflen= " + BufSize + 
  " nbuf= " + NumBuf + " port= " + Port);
Sock = Server.accept();
report("Receive connection from:\n " + 
  Sock.toString() + ");
in = new DataInputStream(Sock.getInputStream());
starttime();
for (i = 0; i < NumBuf; i++){
in.readFully(buf, 0, BufSize);
}
) catch (EOFException e) {
  report("Receive: EOF");
) catch (Exception e) {
  report("Receive: socket or IO error: " + e);
} finally {
  endtime();
  savestats(i);
  try {
    if (in != null) in.close();
    if (Sock != null) Sock.close();
    if (Server != null) Server.close();
  ) catch (Exception e) {
    report("Receive: socket close error: " + e);
  }
in = null;
Sock = null;
Server = null;
}
...

One thing I found interesting is that sockets have to be explicitly closed if you intend to use them again. My initial interpretation of how Java worked was that terminating the thread would close the socket, allowing me to reuse it. This isn’t the case, and Java reported a socket already in use exception when I tried repeating a test using the same socket.

Other Observations
I enjoyed building the Java version of TTCP (you can download the Java executable from our Web site <http://www.ccc.com>) and learned quite a lot about object-oriented programming in the process. One of the things that I found surprising is the lack of a `getargs()` argument-parsing class. My Web searches found only an example of processing command arguments using in-line code. The differences in compilation speed between the Sparc and Cafe compilers was surprising, especially considering the higher speed of the Sparc 20 CPU. The development environment is useful for rapid prototyping, but has some drawbacks for production use, which I believe will be fixed as the product matures.
MrMean the hacker

Social Engineering System Administrators on the Internet

I write this article because so many system administrators panic when they receive a message saying they are being attacked. Panic upsets lunch and makes everyone cranky in addition to wasting resources and money. I hope to help people deal with “social engineering” attacks, which are a common event on the Internet.

The simplest and bluntest definition of social engineering is a way of obtaining a goal by means of lying or deceit. This may make it sound easy to detect but for some reason when social engineering is combined with technology, people are easily fooled. It is a very effective combination.

The following exchanges took place during May 1997. I changed the userid of the person sending me the email to MrMean, and I removed most of the email headers because I saw no reason to harm whoever this person is. He was unable to harm me, and revenge is sweet only if there is something to revenge. I have not changed the user text in any way except to hide who sent it.

This is the first message received. Notice that the user sent the message to Webmaster, not root or postmaster. He most likely browsed our Web site and hit the Webmaster reply button.

From MrMean@aol.com Tue May 20 21:14:41 1997
To: webmaster@cyber.com
Subject: hackers Status: OR

I am a hacker and if you want to get a program to keep us out of places we cant get in to you will never see your webmaster program will never be there again you ass from MrMean and i am MrMean’s Pal by

The first message was intended to terrify the system administrators. I believe that this was just social engineering. Why tell me that you are a hacker and that you can do nasty things to me when a real hacker wouldn’t want to be found and would just do whatever he wanted? In addition, the systems were all running fine. I decided to draw the “hacker” out and learn just what he was up to. If this was a real hacker my offer to admit he is better than me should be as sweet as honey, and he might tell me where my security hole, if any, is. Finally, the “hacker” is an AOL user. AOL being a commercial Internet Service Provider certainly should know who is using this account, and that information can be obtained by court order. My reply:

From radatti@cyber.com Wed May 21 10:17:42 1997
Subject: Re: hackers
To: MrMean@aol.com

How very clever of you. So, if you got into my system where is the cookie? Why don’t you leave a file called /tmp/hack on my system and tell me all about it. If you really did get in and you tell me how to fix it then I will publish a paper on my web site saying that you got in and proved it.
Keep in mind that we did not really go to too much trouble to secure the system, all we did was install some wrappers and disable a bunch of stuff in the kernel. Mostly we rely on backups.
I look forward to your reply.
Pete Radatti radatti@cyber.com

MrMean wasted no time in sending his reply. In fact, he sent two replies separated by about 30 minutes. Notice that he hit the reply button this time instead of continuing to send to the Webmaster.

From MrMean@aol.com Wed May 21 22:42:42 1997
To: radatti@cyber.com
Subject: Re: hackers

i got in your system and i can prove it because i copied your passwords and i destroyed one of your kernels and if you look a lot more in your system you might find the letter that i wrote mess with the best die like the rest

At this point I am sure that MrMean is not a real hacker, or if he is, he is very young and unskilled. He ignored my reply, didn't take the honey, and blustered too much. I checked the systems. The kernels were all there, and I could not find a message. Lies that are easy to verify are not very effective. Let's see where this will go.

From radatti@cyber.com Thu May 22 09:28:10 1997
Subject: Re: hackers
To: MrMean@aol.com

OK, I am lame. I looked all over the www system for your message and couldn't find it. The kernels are still there. Tell me where to look.

Now MrMean is claiming to be MrMean's mom. I guess it is possible but the real information is contained in the word "spam." CyberSoft has a problem with spammers faking our cyber.com domain. This has cost us thousands of dollars in wasted time and resources and has been the cause of us receiving death threats from people who just don't bother to read our automated reply. If you want to see it, send a message to <remove@cyber.com>. This is also another indication of MrMean's age. Very few hackers will ever claim to be their mom or rely upon parental authority to try to scare someone off.

From MrMean@aol.com Wed May 21 23:13:42 1997
To: radatti@cyber.com
Subject: Re: hackers

Stop sending spam here MrMean's mom.

Because MrMean is now claiming to be an adult, I will treat him as such. Notice that I am using my title, thus conferring the status of at least "equal" to the adult. If MrMean is a juvenile, this puts me in a superior position. This is also the last message that either of us will bother with because the game is over.

From radatti@cyber.com Thu May 22 08:57:59 1997
Subject: Re: hackers and spam
To: MrMean@aol.com

Dear MrMean's Mom,

We NEVER spam. We have never spammed and will never do so. We do get hit by people faking our domain address at least three times per week. When this happens we get flooded with about 17,000 remove messages. If you had sent a message to remove@cyber.com you would know this. When we can find out who faked our domain address we request they stop. If they do not then we press charges.
If you received a spam that appears to have come from the cyber.com domain then please send a copy to us so we can go after the person doing it.
Your son sent us a threatening email saying that he was going to damage our systems. We could have gone to the FBI with such an email but we felt that might have hurt him and we really do care about people, even people who threaten us. If you have any other suggestions, we will be happy to hear them.

Pete Radatti
President CyberSoft, Inc.

Conclusion
Social engineering can be as destructive to an organization as a real attack, and many people just don't know how to handle it. CyberSoft has in place some policies that make dealing with these problems easier:
1. Look for evidence on the systems. This took about 15 minutes using CyberSoft's CIT product. If you don't have CIT and you are using UNIX, run Tripwire. Run COPS and Tiger Script. Examine the "last" log.
2. Print off the messages. If they came by email, include the message headers. Most people do not know how to hide their identity on the Internet. A handle hides nothing because the Internet Service Provider knows who is paying for the account.
3. Let everyone on the network know about the "attack message" so anyone who knows anything will tell you. The other benefit of letting everyone know is they will not panic if contacted. This wastes time, but less time than a panic.
4. Do full backups to off-line media. You should be doing this anyway.
5. Learn more or ignore it. You need to make a judgment call to reply or not. Don't automatically assume that anything the "hacker" tells you is true, but verify for yourself. If you do decide to learn more, then be respectful, and don't push too much, or you may find that your MrMean has friends.
6. Decide where to draw the line. CyberSoft always responds to death threats (no matter how unlikely they may be) by contacting the legal department of the company that originated the message. We may also contact the police, but never the person who sent the message. You need to make a list of things you will always respond to and how.

Finally, the really good hackers don't rely on social engineering except as an accessory. They rely upon their technical skills.

The creation of this article was influenced by Bill Cheswick's famous paper, "An Evening with Berferd, in Which a Cracker is Lured, Endured, and Studied."
using C++ as a better C

C pointers are a major strength of the language, and also a major weakness and source of programming mistakes. Sometimes you really truly need to use a pointer, but at other times, for example, to implement multiple return values from a function, C++ references can provide a superior alternative. This column also looks at a related topic, initialization.

References
A reference is another name for an object. For example, in this code:

```c
int i;
int& ir = i;
```

`ir` is another name for `i`. To see how references are useful, and also how they’re implemented, consider writing a function that has two return values to pass back. In ANSI C, we might say:

```c
void f(int a, int b, int* sum, int* prod)
{
    *sum = a + b;
    *prod = a * b;
}
```

```c
void g()
{
    int s;
    int p;

    f(37, 47, &s, &p);
}
```

In C++, we would say:

```c
void f(int a, int b, int& sum, int& prod)
{
    sum = a + b;
    prod = a * b;
}
```

```c
void g()
{
    int s;
    int p;

    f(37, 47, s, p);
}
```

One way of viewing references is to consider that they have some similarities to C pointers, but with one level of pointer removed. Pointers are a frequent source of errors in C.

A reference must be initialized, and its value (the object to which the pointer is pointing) cannot be changed after initialization. The value of the reference cannot change, but the value of the referenced object can, unless the reference is declared as constant. So, for example:

```c
int i = 0;
int& ir = i;
ir = -19;       // i gets the value -19
```
is acceptable, while:

```c
    const int& irc = 47;
    irc = -37; // error
```

is not. A constant reference that points at a value like 47 can be implemented using a temporary.

References are especially useful in argument passing and return.

**Global Initialization**

In C, usage such as:

```c
    int f() { return 37; }
    int i = 47;
    int j;
```

for global variables is legal. Typically, in an object file and an executable program these types of declarations might be lumped into sections with names like “text,” “data,” and “bss,” meaning “program code,” “data with an initializer,” and “data with no initializer.”

When a program is loaded by the operating system for execution, a common scheme will have the text and data stored within the binary file on disk that represents the program and the bss section simply stored as an entry in a symbol table and created and zeroed dynamically when the program is loaded.

There are variations on this scheme, such as shared libraries, that are not our concern here. Rather, I want to discuss the workings of an extension that C++ makes to this scheme, namely, general initializers for globals. For example, I can say:

```c
    int f() { return 37; }
    int i = 47;
    int j = f() + i;
```

In some simple cases, a clever compiler can compute the value that should go into j, but in general, such values are not computable at compile time. Note also that sequences like:

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

    A a;
```

are legal, with the global a object constructed before the program “really” starts and destructed “after” the program terminates.

Because values cannot be computed at compile time, they must be computed at runtime. How is this done? One way is to generate a dummy function per object file:

```c
    int f() { return 37; }
    int i = 47;
    int j; // = f() + i;
    static void __startup()
    {
        j = f() + i;
    }
```

and a similar function for shutdown as would be needed for calling destructors. Using a small tool that will modify binaries and an auxiliary data structure generated by the
compiler, it’s possible to link all these _startup() function instances together in a linked list that can be traversed when the program starts.

Typically, this is done by immediately generating a call from within main() to a C++ library function _main() that iterates over all the _startup() functions. On program exit, similar magic takes place, typically tied to exit() function processing. This approach is used in some compilers but is not required; the standard mandates “what” rather than “how.”

Some aspects of this processing have precedent in C. For example, when a program starts, standard I/O streams stdin, stdout, and stderr are established for doing I/O.

Within a given translation unit (source file), objects are initialized in the order of occurrence and destructed in reverse order (last in, first out). No ordering is imposed between files.

Some ambitious standards proposals have been made with regard to initialization ordering, but none has caught on. The draft standard says simply that all static objects in a translation unit (objects that persist for the life of the program) are zeroed, then constant initializers are applied (as in C), then dynamic general initializers are applied “before the first use of a function or object defined in that translation unit.”

Calling the function abort() defined in the standard library will terminate the program without destructors for global static objects being called. Note that some libraries, for example, stream I/O, rely on destruction of global class objects as a hook for flushing I/O buffers. You should not rely on any particular order of initialization of global objects, and using a startup() function called from main(), just as in C, still can make sense as a program-structuring mechanism for initializing global objects.

**Jumping Past Initialization**

C++ does much more with initializing objects than C does. For example, class objects have constructors, and global objects can have general initializers that cannot be evaluated at compile time.

Another difference between C and C++ is the restriction C++ places on transferring control past an initialization. For example, the following is valid C but invalid C++:

```c
#include <stdio.h>

int main()
{
    goto xxx;
    {
        int x = 0;
        xxx:
        printf("%d\n", x);
    }
    return 0;
}
```

With one compiler, compiling and executing this program as C code results in a value of 512 being printed, that is, garbage is output. Thus the restriction makes sense.

The use of goto statements is best avoided except in carefully structured situations such as jumping to the end of a block. Jumping over initializations can also occur with switch/case statements.
Are you Tcl-ish?
Summary by Peter H. Salus

I leave the actual blow-by-blow of the Fifth Tcl/Tk Workshop to the rapporteur. Most of this will concern only a couple of sessions.

These opened with a welcome and announcements by the program chairs, Joe Konstan and Brent Welch. Don Libes, having won both last year’s and this year’s Best Paper award, got to introduce the keynote speaker: Brian Kernighan. Don told of having first met him while still in high school and of the supportive influence Brian had had upon him. It was a wonderful intro, revealing a number of aspects of both personalities.

To me, Brian is a treasure of the computing community: where would we be without K&R, without Kernighan and Pike, without Awk, without Kernighan and Plauger? And that’s nowhere near all.

Brian began with a slide of a Sun ad that indicated that JAVA was “Robust” [= Robust] and “Interrupted” [=Interpreted]. “There is a lot of meaningless hype about languages.” From there he went on to “three things I built Tcl/Tk interfaces for”:

- text oriented for a mathematical programming language
- drawing tool
- visualization tool

Brian said that for each problem domain, he would talk about the following:
1. his experience with Tcl/Tk
2. a comparison with VB and Java
3. features/architecture/performance

He said that at the end of his talk he would discuss lessons learned from the “exercise.”

I didn’t mention the book on AMPL but it was published about four years ago (it’s by Fourer, Gay, and Kernighan). Brian now spoke about this language for mathematical optimization problems, which was written “in about 60K lines of C++,” and of the utility of a front-end language and a command-line interface. He got a big laugh for mentioning the possibility of a visual interface, VAMPL++, which could be used, like most interfaces, for “playing Adventure.” With Tcl/Tk, Brian was able to create buttons dynamically so that “every piece of text is live.”

Moving on to VB, Brian said that it was a “dialect” of Basic where “controls” are analogous to widgets. “Boy, is it clunky,” he said. He pointed out that VB has an “integrated design and run environment that is nice . . . except that the text editor part is quite bad.”

In terms of a VB interface with AMPL, Brian said it had “problems with IPC,” that there was an excess of “tedious geometric programming,” that it was “non-parallel, running only on Windows," but that its “high extensibility is a great advantage” (you can buy the Awk widget for $49) – there is thus a large market in third party controls for extending VB.

Java is seen by Kernighan as a “cross between C and C++.” It is platform independent and can be run within a Web browser. AWT is a “large but not powerful library.” “Java provides network access reasonably well." IPC is easy: better than VB, worse than Tcl; file I/O is awful (“slow beyond belief”); the text widget is weak (as is VB’s). The “language is pretty good, the libraries are pretty bad . . . immature,” said Brian.

None of the three stands out as very much better than the others (for this specific task); they all work with reasonable speed, though VB has “size limits.” “The Tk text widget and event bindings are more flexible and yield a richer set of possibilities” than the other two.

After showing the audience a message sequence chart (ITU Z.120), a specialized kind of diagram, rendered by each of the
three languages, Brian noted that VB was “easy to get started, but then you run into a wall,” that VB was “astonishingly slow and unusable,” and “when the PC version of Tcl/Tk came along, I dumped VB.” He said that a Java implementation of the standard of this tool would be hard, too, though easier than VB. This is because “the Tk canvas widget makes it so easy to retain and interact with graphical structures.”

After several further anecdotes, Brian stated that he was “a real enthusiast” where Tcl/Tk was concerned, though it was a “well-kept secret.”

Asking a number of rhetorical questions (“Why do languages succeed? Why do languages fail? Does it matter?”), Brian stated that:

1. Good programming languages matter.
2. Good programming matters more than good languages.
3. No language will solve all our programming problems.

In connection with this, Brian cited the anthropologist Benjamin Lee Whorf. I nearly gasped aloud – not because Whorf is unfamiliar, but because this was the same citation made by Bjarne Stroustrup a month earlier at COOTS, and Bjarne also supported very strongly Brian’s statement in the previous list. The use of Whorf (Carol Eastman [Aspects of Language and Culture, 1975, p. 100] notes that Whorf’s hypothesis tends “to reinforce the idea that each language and culture embodies a particular world view”), whom Brian tells me he has been citing “for at least a decade,” is one thing. The more important point is Brian’s and Bjarne’s belief that there’s no magic programming language: no programming language will be able to come to grips with all the problems of programming.

This means that the future of programming languages may well lie in good tools and little languages, but I still see an important role for “general” languages. Brian expects “it just means that people will continue to invent languages, some of which will be improvements.”

On Thursday morning, there was a panel discussion concerning the future of Tcl/Tk. Mike McLennan opened it by saying that Tcl/Tk “is at a crossroads.” It might be a great success or “it could remain the best-kept secret in computing.” The previous morning we had been treated to John Ousterhout’s view of language politics at Sun. After an hour’s discussion, with many voices from the audience, there seemed to be a consensus that Tcl/Tk needed PR, that perhaps a not-for-profit organization could do this, but that something was necessary.

On Tuesday afternoon, Don Libes had several pithy comments:

1. FAQs are indications that the primary documentation has failed.
2. No one ever died from too much documentation.
3. More and better documents are needed.
4. Standards are a myth; I only wish POSIX was.

So do most of us, Don.

In hope of moving Tcl/Tk from its status as a secret, a Tcl/Tk Consortium is being established: <http://www.tcltk.com/consortium>.

Fifth Tcl/Tk Workshop
BOSTON, MA
July 14-17, 1997

Summaries by Brian Bailey

TECHNICAL PROGRAM

Tcl in AltaVista Forum

David Griffin, AltaVista Internet Software Inc.

Dave Griffin presented the first paper of the Fifth Annual Tcl/Tk Workshop on Alta Vista Forum, an award-winning collaboration environment built using the Tcl language. During the initial development of the forum, two main goals were pursued. First, the products built must work within the framework of the World Wide Web. Second, the products built must match the rapid pace of evolution expected in this space. The Tcl language was chosen for the implementation of the forum for the following reasons:

- Extensibility. Several new commands as well as a “mailbox” object were added to the core Tcl language. The extended language provides a custom scripting language for developing Web-based collaborative applications.
- Maturity. Tcl has few bugs and is well documented.
- Portability. Tcl runs under Macintosh, several UNIX variants, and Windows.
- Rapid development. Ideas could be quickly implemented and tested.

Because Alta Vista Forum is a Web-based application, it needs to interact with hypertext servers as CGI applications and generate results using the HTML language. The forum is controlled by three executables:

- Dispatcher. A CGI application, essentially an extended Tcl interpreter, that analyzes the request, loads the application, and executes the appropriate Tcl
scripts, which ultimately generate the HTML response

- Butler. Similar to tclsh, but with most of the extensions of the dispatcher.

- Background. Executes butlers with small Tcl scripts

One of the initial problems with Alta Vista Forum was performance. To overcome the performance problems, two methods were used. First, a new facility called Tcl Package Libraries (TPL) was created. All of the Tcl code was placed in a single, cross-platform archive that can be read in by the various executables as needed. Second, persistent servers were used. Through instrumentation of the CGI process, a great deal of time was being spent creating and initializing the dispatcher process. To reduce this overhead, the idea was to have a dispatcher process resident in memory waiting for a transaction. Once received, the transaction would be analyzed and processed and a result generated. Finally, the dispatcher process would be reinitialized and wait for the next transaction. Because the dispatcher is essentially a Tcl process, a triad of interpreters was used to meet these requirements. The three interpreters used were:

- Master. Maintains the initialization state of the dispatcher

- Transaction. A transient slave interpreter created by the master for each new transaction

- Pristine. Slave interpreter created by the master for each application class

For small transactions, this architecture resulted in a 5x performance increase.

Overall, the developers of Alta Vista Forum were happy with the Tcl language, but they would like to see the following in future releases:

- Syntax checker
- Debugging tools
- Profiling tools

"Dashboard": A Knowledge-Based Real-Time Control Panel

De Clarke, UCO/Lick Observatory

De Clarke presented the second application-oriented paper on "Dashboard," a user interface for the DEIMOS instrument located at the UCO/Lick Observatory. First, some essential background information was given on how astronomers handle data. Astronomers use a data storage convention known as FITS, essentially a set of keyword/value pairs, for the archival and interchange of image and tabular data.

During the early planning stages for the DEIMOS instrument, the software team realized they would need to manage a large number of new keyword/value pairs. Thus, a relational database was constructed for modeling FITS keywords and storing keyword attributes such as data type, format, read/write access, and semantics. When completed, the keyword database became a powerful resource from which they could generate documentation, sample FITS headers, and certain repetitive sections of Tcl/Tk source code.

Another requirement was a good graphical interface for bench tests, development, and preship qualifications. The goal of the user interface was not a specific, hand-crafted product tailored for DEIMOS, but a generic, "soft" application capable of reading the keyword database and configuring itself accordingly. To do this, each keyword could have an associated set of Tcl/Tk source code. In other words, the database was not merely used as the target of the application, but as an integral part of the software design, configuration, and deployment.

Each keyword in the database can be linked to a global Tcl variable. If the keyword is being monitored, the KTL control software sends out a broadcast message each time the value portion of the keyword/value pair changes. The associated Tcl variable is then updated with the new value. That variable can then be used as the "-textvariable" option available with most Tk widgets (thus, as the database is updated, the widgets reconfigure themselves) or can be associated with an arbitrary Tcl script (presumably through the Tcl trace facility).

The "dashboard" is a single application, with one maintenance cycle and one investment in development, which can be used to provide engineering diagnostic interfaces for any number of instruments sharing the KTL control protocol. Nothing restricts the use of the "Dashboard" application to astronomy. Any keyword/value pair control system could use the "Dashboard" code with little modification.

Caubweb: Detaching the Web with Tcl

John R. LoVerso and Murray S. Mazer, Open Group Research Institute

John R. LoVerso presented the third and final paper of the first Applications session. This paper deals with the problem of providing Internet resources to the user even when the user is not currently connected to the World Wide Web (e.g., on an airplane). The problem is complicated by the need to provide the user with both read and update access to disconnected information. Caubweb is the system developed to address these issues.

One of the main goals of the Caubweb system was to be browser independent (i.e., not to be tied to any off-the-shelf browser such as Netscape or Internet Explorer). As a result, Caubweb was designed as middleware, sitting between the local browser client and the hypertext server. Users simply start the Caubweb system, configure their client browser to proxy through Caubweb, and browse the Web as usual. Caubweb acts as a transparent "middleman" in the browsing activity by caching all information that passes through it. However, the user may also explicitly declare other information to be cached by defining "weblets," con-
nected subsets of Web content meeting user-defined criteria. Weblents are specified through qualification predicates that Caubweb applies to documents. The application of the qualification criteria to the documents occurs asynchronously so as not to interrupt the user's normal browsing.

Once disconnected, Caubweb impersonates those servers on which the cached information actually resides, thus seemingly providing Web access even though the user is currently disconnected. Once reconnected, published pages are written to their proper location, and any missed information is retrieved and cached for the next disconnection period.

The entire Caubweb system was built using the Tcl language. Tcl was chosen for the following reasons:

- Ease of programming. The learning curve for developers to produce good, usable code is short.
- Portability. Tcl code runs under several UNIX variants, Windows, and Macintosh.
- Extensibility. Shared libraries can extend the language.

However, Tcl did have its shortcomings:

- Event-driven architecture. Caubweb used an internal architecture of asynchronous execution with callbacks. Any operation that has a long computation time appears to "block" the application. In other words, the application is non-responsive until the callback finishes and control is returned to the main event loop.
- Performance. The interpreted environment of Tcl proved too slow in some situations.
- Extensions. Many were not portable to other platforms and did not support some basic language operations.
- No standard library. Tcl is missing a rich, cohesive, organized, and standard extension library.

The latest version of Caubweb is available from <http://www.opengroup.org/RI/WWW/DIST_CLIENT/caubweb>.

Jacl: A Tcl Implementation in Java
Ioi K. Lam and Brian C. Smith, Cornell University

Ioi Lam was the recipient of the Best Student Paper award for the Fifth Annual Tcl/Tk Workshop. Ioi Lam began the Implementation Issues session with his presentation on Jacl. Simply stated, Jacl is a Tcl implementation in Java. Implementing the Tcl interpreter using Java provides several benefits:

- Portable extensions. Currently, Tcl extensions are written in C, which makes them difficult to port to different platforms. If extensions were written in Java, they would automatically be portable to any platform with a Java Virtual Machine running.
- Robust Java scripting language. According to the authors, Java needs a scripting language as powerful as Tcl. Other scripting languages, such as JavaScript and VBScript, are proprietary, nonportable, and tied to specific browsers.

The current status of the project is that all core Tcl commands have been implemented and tested, and work on the Tk widgets is underway. Although little Tk support is currently available, a short animation showing text scrolling across several buttons was demonstrated. Writing the Jacl interpreter and extensions in Java creates an embeddable,universal scripting language for Java.

A Typing System for an Optimizing Multiple-Backend Tcl Compiler
Forest Rouse and Wayne Christopher, IECM CFD Engineering

Forest Rouse presented the typing system used within the ICE 2.0 Tcl compiler. One of the advantages of using Tcl is the lack of type declarations (everything is a string), but for a Tcl compiler, lack of typing makes optimization very difficult. The ICE compiler was first designed to emit only C code, but now it has been extended to emit either Tcl 8.0 bytecodes or C code. The approach taken is that the compiler first translates the Tcl code into an intermediate language representation and from this representation generates either Tcl 8.0 bytecodes or C code.

The typing system can be thought of as a set of attributes whose values at any point in the program represent the current Tcl type that is associated with every recognized Tcl variable. Unfortunately, this is a difficult task due to the inherent side effects allowed by the Tcl language. Thus, the typing system must not only maintain the type of the variable, but also its locality (global, local, upvar, uplevel or argument), whether the variable has been unset, current traces, and more. Essentially, the typing system assumes the worst case scenario and recomputes these attributes upon every state change of the system. However, users can constrain the effect of statements through the use of "promises." Promises are made regarding the possible side effects of certain Tcl commands.

Performance measurements on the final version of the ICE 1.0 compiler have shown a factor of 7.5x speedup, whereas the Tcl 8.0 bytecode compiler from Sun has shown a factor of 4 speedup on the same benchmark. To get even better performance increases, other optimization techniques are necessary and will require an advanced typing system.

TCLOSAScript – Exec for MACTcl
Jim Ingham, Lucent Technologies and Raymond Johnson, Sun Microsystems

Jim Ingham presented the final paper of the Implementation Issues session on an exec mechanism for the Macintosh. On UNIX systems, parent processes can communicate with spawned processes by redirecting the child's stdin and stdout.
channels to channels of the parent. The “exec” Tcl command leverages this capability to invoke system commands, e.g., the UNIX date command. Unfortunately, Macintosh computers do not have an equivalent concept of a command line or stdin and stdout. Thus, an alternative method for implementing the exec command must be realized. That was the focus of this presentation.

Macintosh provides the Open Scripting Architecture (OSA) for interprocess communication. Essentially, every OSA-compliant application adheres to a standard set of “events” (e.g., get, set, and create) as well as a set of application-specific events (e.g., select in a word processor). These events are very low level, and incorporating them into a higher level language such as OSA is beneficial. The good news is that OSA can be used to implement the Tcl “exec” command.

Three main steps are required to fit Tcl into the OSA architecture:

- Allow Tcl to use the services offered by other OSA-compliant components installed on the system. This is the function of the TcOSAScript extension that has been completed. In fact, TcOSAScript will be included in the 8.0 release of MacTcl.
- Provide a Tcl command to build and dispatch Apple Events. This step has not yet been completed.
- Install Tcl/Tk as an OSA-compliant component in its own right so that Tcl would be available to other components. This step has also not yet been completed.

This work should allow Tcl to take advantage of and offer its own services to other OSA-compliant applications.

Redesigning Tcl-DP
Mike Perham, Brian C. Smith, Tibor Janosi, and loi K. Lam, Cornell University

Brian C. Smith started off the retrospective session with his work on redesigning Tcl-DP. The original Tcl-DP implementation was released four years ago, but upgrading it to newer Tcl releases was increasingly difficult, plus several of its features had already been incorporated back into the Tcl core. Thus, a new Tcl-DP version (4.0) has been rewritten from scratch. Tcl-DP 4.0 includes communication support for serial links, IP-multicast, TCP, UDP, email, RPC, and also allows for filters.

Generally speaking, filters transparently modify data received on a channel before the application sees it. Two types of filters are available:

- Plugin filters are designed for the common case where the data are modified using a functional interface. An example of a plugin filter is the encryption/decryption of data.
- Filter channels allow the creation of modified network stacks (e.g., dataframes over a streamed channel) simulate UDP over a TCP connection.

The new RPC mechanism includes the following enhancements:

- Recursive and out-of-order calls. DP creates activation records for each RPC sent. The activation records provide a convenient mechanism for tracking which RPCs have been received and which are still outstanding and processing them in the correct order.
- Multiple channel support. RPCs can be made over any Tcl channel as long as it has been registered with the dp_admin command.

In previous versions of DP, “peeking” was allowed on socket reads. Peeking enables returning a portion of the data waiting to be received on a socket, without actually consuming any of the data. During the development of DP 4.0, Tcl’s buffering of all channel input made peeking on channels impossible. Dp 4.0 provides a workaround through the dp_recv command. This command allows direct, unbuffered access to a channel’s input procedure.

Tcl-Dp 4.0 is freely available and can be obtained from <http://simon.cs.cornell.edu/Info/Projects/zeno/Projects/Tcl-DP.html>.

Writing a Tcl Extension in Only Seven Years
Don Libes, NIST

Don Libes was the recipient of the Best Paper award for the Fifth Annual Tcl/Tk Workshop. The award paper was a retrospective on Expect, one of the first extensions ever built for the Tcl language. Expect is an excellent candidate for a retrospective because of its long history and popularity. (Expect is used by hundreds of thousands of companies and institutions around the world.) Don described his experiences (both good and bad) resulting from maintaining this extension throughout several years of Tcl evolution.

Expect is a tool for automating interactive applications such as Telnet, FTP, passwd, rlogin, etc. Much of the flexibility of Expect is due to the Tcl language. Tcl provides the basic infrastructure for variables, procedures, and expressions, thus allowing Expect to focus specifically on its function: automating interactive processes. However, even though Tcl at first appeared to fit well with Expect, it did have its drawbacks:

- Lack of null string support. Tcl strings have never supported the null character, but some interactive programs (e.g., curses) actually do send and receive nulls. Thus, Expect had to provide this support itself. Fortunately, Tcl 8.0 will support nulls; unfortunately, seven years have passed since the initial request.
Pattern strings. One of the difficulties with Tcl is learning the quoting conventions. However, compared to other scripting languages, Tcl quoting is easy to learn and a pleasure to use. Another interesting aspect of Tcl is the expr command. The current expr command is now much more like a traditional programming language, but unlike anything else in Tcl.

Name collisions. The first release of Tcl had no file I/O and therefore no open or close commands. Because these command names were not already taken, Expect used them. Once Tcl added file I/O and used these names for its own purposes, name collisions resulted. Unfortunately, Tcl provides no clean solution, and extension writers must deal with this problem themselves.

Event notification. Not until Tcl 7.5 did the Tcl core provide event management. Thus, whenever Tk was not present in Tcl versions prior to Tcl 7.5, Expect had to provide its own event handling. Once Tcl did provide its own event handling, Expect had to make a few (to put it mildly) changes.

Portability. Tcl is portable; thus, Tcl extensions must also strive for portability, but this is not always a trivial task. Most people think POSIX has solved all of the portability problems, but it hasn't. Don pointed out that POSIX doesn't do any good on pre-POSIX systems and that POSIX does not standardize everything. Pys, or pseudoterminals, are a good example of something POSIX does not cover, but was needed in Expect.

Configuration. Tcl provides its own configuration mechanism, but leaves the extension writers essentially on their own.

Documentation. Extension writers are encouraged to document their work through man pages, online documents, classes, papers, and books.

At the end of his presentation, Don was asked whether or not he would use Tcl if he had to write Expect all over again. Don replied that he would have to think about it.

Simple Multilingual Support for Tcl
Henry Spencer, SP Systems

Henry Spencer began the second day of the workshop with his presentation on how to retrofit multilingual support into existing Tcl applications. Even though an application may be portable across different computers, this doesn't mean that it is also portable across national, linguistic, or cultural boundaries. One of the obvious problems is the language hard-coded inside the application, including the user interface, error messages, and command prompts.

Most applications needing multilingual support use the concept of a message catalog. Inside the application, hard-coded words are replaced with message identifiers that are used as an index into the message catalog. Multilingual support is then achieved by creating different message catalogs, one for each language.

The system designed by the author, called Transit, works in a similar, but unique, way. Transit provides a wrapper around the familiar puts command as the primary programming interface. This new wrapper translates anything sent to stdout, stderr, or stderr and leaves everything else alone. The identifier, or key, used to index the message catalog is the original text used in the program. Benefits of this method include the following:

- It eliminates the need to define a new key space. No set of identifiers needs to be created.
- It provides a message of last resort. If the key lookup fails, the original message, even though it may be in the wrong language, can still be printed. The argument is that a message in the wrong language is still better than no message at all.
- It makes the program easier to read, write, and test.

Thus, the Transit equivalent of

puts "hello, world"

is

.puts "hello, world"

However, the author pointed out that it is not always this easy. Tcl programmers tend to build up strings from multiple parts, and sometimes only a portion of the final string should actually be translated. For example, consider an error message that includes the message text itself and a filename; everything but the filename should be translated. The solution was to mark substrings within the overall string and parse the markings at translation time. Literals, such as filenames, are marked with single quotes, and translated substrings are marked with double angle brackets.

The author pointed out that experience with Transit has been limited, but so far the method developed seems to be working and is easy to retrofit into existing applications.

Assertions for the Tcl language
Jonathan E. Cook, New Mexico State University

In general, assertions help programmers write robust code by offering dynamic checking of program properties and enhancing testability. The Tcl assertion package, called AssertTcl, provides the following Tcl commands:

- Assert allows point assertions to be specified about the computation state of the program. These assertions can be placed anywhere in the code.
- Assume allows requirements for procedure input values to be made. Because parameter variable values can be changed anytime in a procedure, new variables of the form param_in, where
param stands for the parameter name used, are created. These variables are initialized to the values of the parameters when the procedure is invoked and should be used within the assume assertion. This method allows the assume assertion to be placed anywhere in the procedure body, but convention says that the assertions should be the first command of the procedure body.

- Assure allows requirements for the return value of a procedure to be specified. Analogous to the assume assertion, a variable, named return_val, will be set to the return value just before the procedure returns. Thus, programmers can use this variable, return_val, inside the assure assertion.

- Always allows guarantees about part of the variable space of the program to be specified. Always takes a list of variables and an expression to be evaluated anytime one of the variables change value.

During development, assertions are a valuable programming tool, but once the application is finalized, assertions may hinder performance. Thus, AsserTcl also allows for the disabling of assertions at the global, procedural, or variable level.

The commands assert, assume, and always were fairly straightforward to implement, but assure was more difficult, primarily because of the semantics of the Tcl return command. To implement the functionality of assure, the procedure body is first parsed, and all return statements not beginning on their own line are replaced with an additional procedure call. The problem is that return is also a mechanism for generating exceptions, which must be passed to the caller of the procedure. If the original return statements are replaced with an additional procedure call, the behavior of an exception has been changed (it's now passed to the caller, when it should be passed to the caller's caller). However, as long as return is used simply to return a value, assure works without any pitfalls.

Finally, a few suggestions for Tcl were given:

- Extend the exception-generating mechanism of the return command so that it is possible to throw the exception up to a specified level.

- Extend the info command to include more context about the current execution point (e.g., the current line number, procedure name, and filename).

Extending Traces with OAT: An Object Attribute Trace Package for Tcl/Tk
Alex Safonov, Joseph A. Konstan, John V. Carls, and Brian Bailey, University of Minnesota

Alex Safonov presented his work on extending the Tcl trace facility to Tk widgets and other Tcl extensions. The Tcl trace mechanism allows Tcl programmers to have scripts executed when a variable is read, written, or unset. Using traces has proven to be useful for several applications the presenter has written, but unfortunately, traces currently work only on Tcl variables and do not apply to objects resulting from other Tcl extensions. The following are the goals of this work:

- Extend the current Tcl trace facility to Tk widget attributes.

- Provide a general mechanism for extension writers to allow their objects and associated attributes to be traced.

- Separate change detection from constraint management.

- Make the extended trace facility easy to integrate with existing Tcl/Tk installations.

Alex illustrated the usefulness of both traces and constraints involving Tk widgets with a short demo. The demo showed the programming ease in which one can keep circles and lines connected on a Tk canvas.

Alex stated that two types of users should benefit from this work:

- Tcl script programmers. By using the extended trace facility, script programmers' resulting code will be more compact and easier to maintain.

- Tcl extension writers. By adding their objects to the extended trace facility, extension writers can allow script programmers to set traces on their own objects in a consistent and well-defined manner.

As a result of having two sets of potential users, the OAT protocol consists of two parts:

- Tcl API, used by Tcl script programmers for creation, deletion, and querying of extended traces

- C API, used by extension writers to extend the trace facility to their own objects

Because traces are essentially a low-level change detection method, they lack higher-level expression mechanisms. Thus, TclProp has been extended to take advantage of the new OAT interface. Together, constraint formulas can easily include Tcl variables, Tk widgets, CMT clocks, and any other Tcl extension that has been OAT-enabled.

A Tk OpenGL Widget

Unfortunately, the presenter of this paper was unable to attend the conference. No summary is available as a result.

The ImageTcl Multimedia Algorithm Development System
Charles B. Owen, The Dartmouth Experimental Visualization Laboratory

Charles Owen began the Multimedia and Graphics session with his presentation on ImageTcl, a multimedia algorithm development system. Multimedia algorithm
development can be described as a five-step process:
- Devise a theoretical algorithm.
- Implement a prototype.
- Devise test procedures for the prototype.
- Test algorithm performance and effectiveness on large data sets.
- Develop user interfaces to support interaction.

The primary goal of the ImageTcl system is to simplify steps 2 through 5. Other multimedia development environments have also focused on providing high-level scripting tools to eliminate the modify-compile-link-test cycle. However, these systems assume that a complete set of tools can be provided without compromising performance, whereas the approach taken in the ImageTcl system is that this is difficult, if not impossible.

The design of ImageTcl can be divided into an execution structure and a development structure. The execution-processing structure is a data flow graph consisting of objects and associated modules. Objects can be thought of as generic processing steps that must be performed, and modules can be thought of as the specific algorithm employed to perform that processing step. The power of ImageTcl is leveraged when several modules (algorithms) exist to perform the same generic processing step. These modules can easily be substituted for one another, tested, and compared. To further facilitate testing of the algorithms, ImageTcl also provides a media database useful for accumulating standard test data for a particular development site. Construction of the graph and all control is implemented in Tcl.

ImageTcl components can be divided into three sets:
- Core - components necessary for system operation
- Standard - components included in the published system.
- User - one of the more important features of ImageTcl. Users create new commands, objects, or modules through a set of fill-in forms built using Tk. Once described to the system, template header and source files are automatically generated, and the user must fill them in. Once filled in and compiled, the component can immediately be used in the system.

ImageTcl has been or is currently being used in the following projects:
- Functional Magnetic Resonance Imaging data analysis
- Text-to-speech alignment.
- Cut and pause detection

Nsync - A Constraint-based Toolkit for Multimedia
Brian Bailey and Joseph A. Konstan,
University of Minnesota

Brian Bailey presented on a multimedia synchronisation toolkit written entirely in Tcl. In general, multimedia applications can be partitioned along three axes:
- Media. The audio, video, animation, or text used within the application
- Synchronization. The "glue" one uses to assemble the different media components
- Interaction. The control the user exercises over both media and interaction

The Nsync toolkit focuses on providing synchronization and interaction while reusing an existing toolkit, the Continuous Media Toolkit from Berkeley, to provide media content. The designers set the following goals for the Nsync toolkit:
- Application neutral. Nsync is not limited to a specific application domain (e.g., hypermedia or slide presentations). The idea is to provide a basic mechanism so that these types of applications can easily be built using the toolkit.
- User interaction incorporation. The toolkit directly allows user interaction to be specified.
- Programming effort minimization. Toolkits should reduce the complexity of tasks that are normally difficult, like defining coordination of events through time.
- Useful for nonmultimedia applications. Because the toolkit controls media streams through the logical time system of CMT, any application using that same notion of time can also benefit from the Nsync constraint mechanism.

Within the toolkit, media are assembled by first defining a temporal expression and then associating an action along with it. Whenever the temporal expression becomes true, the associated action is invoked. At first this seems very simple, but because one or more logical clocks (i.e., variables representing the current logical time) can be included in the expression, standard expression evaluators are useless. Nsync provides its own temporal logic evaluator, which can evaluate both temporal and nontemporal expressions and, for temporal expressions only, predicts when and if any logic transitions will occur. By making the correct predictions and invoking the associated actions at the appropriate times, the toolkit keeps applications "in-sync."

In order to better understand how one uses the toolkit, Brian presented an example that dealt with a timed slide sequence of a virtual house tour. The essence of the example was that the user should be able to exercise some control over the transitioning of the slides, rather than only letting the system perform the transitions at the predefined times. Through the use of a button, the user could "hold" any slide in place and not allow a slide transition to occur. Brian showed how two simple temporal expressions, along with their associated actions,
could implement all the necessary requirements.

At the end of the presentation, Brian gave a Tcl wish list:

- Multithreaded core. Because media playback is implemented as a set of callbacks for the main event loop, anytime the event loop goes off to handle other events (e.g., button presses), media playback must be discontinued. These interruptions may be very noticeable, depending on the length of time the event loop is off handling other callbacks.

- Realtime after queue. Although the predictions made by the temporal logic evaluator are fairly accurate, the Tcl after command makes no guarantees about the requested versus actual invocation times.

- Variable type identification. Sometimes it's necessary to know the "type" of object that a variable contains. (e.g., whether the variable represents a logical time line or just an integral value).

Nsync can be obtained from [http://www.cs.umn.edu/Research/GIMME/Nsync.html](http://www.cs.umn.edu/Research/GIMME/Nsync.html)

Managing Tcl's Namespaces Collaboratively
Don Libes, NIST

Don Libes, who also won the Best Presenter of the Fourth Annual Tcl/Tk Workshop, started off the Development session by presenting a method for collaboratively managing Tcl's namespace. In all programming languages, identifiers are used to identify things (e.g., procedures, variables, or tags.) Choosing an identifier is easy; choosing an identifier that no one else has already chosen is much more difficult. With its single, global namespace, Tcl is very susceptible to namespace collisions (developers choosing the same command or variable name). Extension writers have little or no knowledge about identifier choices made by other extension writers. In the past, the solution to this problem was for extension writers to simply prefix all commands and global variables with a string derived from the extension being written. This reduces the possibility of a namespace collision, but does not completely solve the problem. The solution presented here is to use NICS (NIST Identifier Collaboration Service), a Web-based collaborative registry service designed specifically for identifiers. NICS provides the following features:

- Identifier registration. Users (extension writers) may register identifiers well before the extension is actually shipped. During registration, users would include a brief description of the identifier, current status, and more.

- Identifier comments. Users may privately or publicly comment on other identifiers.

- Instantaneous availability. Because the information is maintained in an online database, access is immediate.

- Conflict notification. If requested, conflicts and comments can be sent to the registered owner of the identifier.

Although the Tcl community has pre-announced support for multiple namespaces, which would also directly address the namespace collision problem, these namespaces must also be named. Thus, the problem has once again been reduced, but not eliminated, because extension writers might choose the same name for different namespaces. Furthermore, NICS would also be useful to the Tcl community to help manage the following:

- Package names
- Tcl completion codes
- Math function names
- Library names

Don also briefly described the user interface to NICS. The interface is based on HTML forms, is very straightforward, and is accessible from any Web browser. Of course, the CGI scripting and database service was developed using Tcl.

The NICS system has been prototyped and is available to a few select communities, including the Tcl community. This represents an opportunity for the Tcl community to both maintain its identifiers and to impact the development of NICS through feedback.

PtTcl: Using Tcl with Pthreads
D. Richard Hipp, Hwaci Corporation

D. Richard Hipp presented on a multi-threaded version of Tcl. Tcl was originally designed to be used in single-threaded applications, but as multi-threaded applications become increasingly popular, so too is the need to use Tcl within them. Unfortunately, enabling multiple threads to be active within the Tcl core is extremely complex due to the large number of static data structures and the need for multiple stack support (recall that typical thread models allocate separate stack space for each running thread). PtTcl is an attempt to provide thread support without completely redesigning the Tcl core. The idea behind PtTcl is to allow zero or more interpreters to run in independent threads. The execution model can be summarized as follows:

- A single thread can have any number of Tcl interpreters.
- Any one interpreter can have only one thread active within it.
- Each thread has its own event queue.
- Threads can communicate back and forth through messages, a new kind of Tcl event.
- Tcl variables can be shared among the different interpreters.

The PtTcl package implements two new Tcl commands, shared and thread. The shared command is used to designate variables that are to be shared among several interpreters. One of the draw-
backs of shared variables is that traces cannot be placed on them. This is a consequence of the fact that an interpreter can be used in only a single thread. For example, if thread A sets a trace on a shared variable and then thread B writes the shared variable, there is no mechanism for thread B to invoke the trace script in thread A. The thread command is used to create and control threads. The thread command can take nine different subcommands, which can be used to create new threads, send and receive messages, or query a thread's state. Each of these commands has both a Tcl and a C interface.

PtTcl was developed under Linux using the MIT Pthreads library. It is currently being used within a multiprocessor industrial controller and has survived extensive abuse testing.

The latest source code for PtTcl can be obtained from <http://users.vnet.net/drh/pittcl.tar.gz>.

A Tcl-based Self-configuring Embedded System Debugger

Dale Parson, Paul Beatty, and Bryan Schlieder, Bell Labs Innovations for Lucent Technologies

Dale Parson presented on TEEM (Tcl Environment for Extensible Modeling). Dale belongs to a group at Bell Labs responsible for building software generation tools, such as compilers, assemblers, emulators, and debuggers, for a variety of embedded digital signal processors. Some processors vary at the core architectural level; others differ only with respect to I/O circuitry or memory configuration. Tcl provides the mechanism for separating processor-specific details from the debugging environment of the simulation and emulation tools.

TEEM is a Tclsh binary extended with processor modelling commands that fall into the following categories:

- Model management. The `psr` command queries model types, constructs processor instances, and deletes model instances. This command gives access to a dynamically linked library of C++ constructors. Each constructed object returns a unique instance name to Tcl which then becomes a new Tcl command name until the object is destroyed.

- Model access. These commands initialize, inspect, and modify model instance state (registers, memory, buses, and pins). Model access is provided through the `expr` command. `expr` syntax is a superset of Tcl's `expr` command. The `?` command includes options for determining the names, types, and properties of user-accessible elements within a model.

- Model control. Execution and breakpoint commands drive the simulation at the C++ level until a breakpoint or exception occurs. The `reset` command resets the current processor's state, and the `step` and `resume` commands advance its state.

- Model I/O. This mechanism allows files or callback procedures to be attached to I/O activity. For example, a processor input action from an I/O port can cause a call to a Tcl procedure that returns a value for that port.

TEEM's Tcl interpreter provides an ideal environment for batch execution, procedural extension, and regression testing. Instead of including GUI code, TEEM provides a generic socket interface to allow client applications to submit Tcl queries. Thus, remote GUIs can retrieve and update model state through this socket interface.

In some situations, the context in which a Tcl_Eval is executing must be maintained. One such situation is when a model initializer encounters a `step` or `resume` command. Unfortunately, if Tcl_Eval is allowed to return, the context in which it was executing is lost. Thus, in these situations, Tcl must not be run as a subroutine, but as a coroutine peer of the host simulator. The coroutine method runs the Tcl interpreter in its own thread, while the simulator thread is blocked. Whenever a `step` or `resume` command is encountered by the Tcl interpreter, the Tcl thread is blocked and the simulation thread is resumed.

GeNMSim - The Agent Simulator

Udi Margolin, Ilana Gani-Naor, and Raz Rafaeli, Milestone Software & Systems, Ltd.

Udi Margolin presented GeNMSim, a Tcl/Tk-based multiprocessor SNMP agent simulator. First, some background information was given on network management systems. A network management system provides the functionality and tools to centrally manage a communication network. Any new equipment added to the network needs a management application that communicates with the network system that is already in place. Thus, a network vendor usually provides the new network equipment, a management application to manage the new network equipment, and a software agent that communicates with the management application. An important part of building a new network device is the design of the management information that will be available for this device. SNMP, the most prevalent management protocol, defines a syntax for this management information called MIB.

In order to develop a management application for a new network device, one needs the device's MIB definition and a working agent. Because a networking device cannot be shipped without a management application, starting the application development only after the agent is already functional causes a substantial delay in the availability of the equipment with its management application.

Using an agent simulator can help reduce this delay by turning the development
process of the network device and the management application into concurrent processes. Using an agent simulator can also help catch bugs in the MIB early in the development cycle, when the problems are easier to fix.

Tcl was used for this project for the following reasons:

- **Portability.** GeNMSim is designed to be a portable product for UNIX and Windows platforms.
- **Ease of development.** Using Tcl saves substantial R&D time. Without the standard compilation cycle of compiled languages, Tcl code changes can be tested and integrated very quickly.
- **End-user customization.** End-users can customize the simulator using Tcl callbacks.

The **Tycho User Interface System**

Christopher Hylands, Edward A. Lee, and John Reekie, University of California, Berkeley

John Reekie gave the final presentation of the Fifth Annual Tcl/Tk Workshop on Tycho, a user interface system for the Ptolemy project at Berkeley. Ptolemy is a large C++ software package used to design, simulate, and generate signal-processing and communication systems. Ptolemy was started in 1990, and version 0.7 is slated for 1997. Just as Ptolemy supports the creation of multiple semantic models, Tycho aims to support the rapid construction of user interfaces to support those semantic models or to visualize application-specific design information. More broadly, the goal of Tycho is to be an extensible framework in which tasks such as documentation generation, indexing, font management, color management, and dialogs with the user are built using a shared, common infrastructure.

One of the features of the Tycho user interface system is it model-view architectural pattern. Tycho has a model class that provides a publish-and-subscribe mechanism, unbounded history, and a simple external file format called TIM (Tycho Information Model). The model class can be subclassed to provide application-specific models such as storage user preferences.

TIM is a meta-data format intended to encourage clean representations of data both in memory and in an external file format. Because the model supports an unbounded history, user interfaces can easily support multilevel undo/redo operations. Each method that changes a model is required to return a script that will undo that change.

Another pattern used in Tycho is the Displayer-View pattern in which one or more views can place themselves in a particular top-level window. This pattern allows the creation of widgets that can be placed into Dispalyers in new combinations.

Although Tycho was originally intended to serve only as a user interface to the Ptolemy project, it has become more. Tycho already includes the following:

- **Collection of mega widgets, including file browsers, font and color selection, and alert boxes.** lwidgets (widgets built from [incr Tcl]) were not used because the Tycho project started before lwidgets were robust enough for heavy use in the Tycho system.
- **HTML browser widget**
- **Finite-state machine editor**
- **Class hierarchy viewer**
- **Interface to the Tcl profiler from the TeX package**
- **Hierarchical canvas items**
- **Interactors (capture patterns of interaction)**

Integration of Tycho with Java is also being explored by this group. The Tycho-Java interface is based on Sun's experimental Tcl-Java interface. The goal of this effort is to use [incr Tcl] / [incr Tk] for the user interface and Java for the back-end processing. The argument for this split is that Tk has a much more mature and flexible user interface package than does Java. Based on experience with the Tycho system, the most important efforts of the Tcl community should be:

- **Provide adequate support for object-oriented extensions to Tcl, such as [incr Tcl]**
- **Provide a seamless and efficient interface to Java**

Tycho is freely available at <http://ptolemy.eecs.berkeley.edu/tycho>.

---

Conference Summarizers Wanted

**;login:** is looking for people planning to attend either of two upcoming winter events – the USENIX Symposium on Internet Technologies and Systems (December 8-11, 1997) and the 7th USENIX Security Symposium (January 26-29, 1998) – who are interested in writing summaries of technical sessions for coming issues. You should be willing to commit to covering at least two sessions.

Information about USITS and the 7th Security Symposium can be found on the USENIX Web site at <http://www.usenix.org/events>.

Please contact ;login:'s Managing Editor, Eileen Cohen, <ecohen@usenix.org>, as soon as possible if you would like to contribute.

USENIX thanks you!
An Update on Standards Relevant to USENIX Members

by Nicholas M. Stoughton
USENIX Standards Liaison

A Message from the Chair
Lowell Johnson <3lg@rs1.unisys.com>, Chair of PASC, reports at the July 1997 meeting in Nashua, NH.

Changes Coming in the Standards Arena

There are several important changes coming in the near future, both to POSIX and IEEE standards in general. POSIX is the more important to this audience; I'll discuss changes to it before the broader changes.

The July meeting of the Portable Application Standards Committee (PASC) Sponsor Executive Committee (SEC) was specifically extended for a special discussion about the future of POSIX standards. This discussion was instigated by a request from HP that, simply stated, requested that PASC no longer be allowed to generate additional POSIX standards. Although ludicrous on the surface, underneath there was some valuable content.

The initial reaction was that this was a cheap shot by HP to save money by reducing the number and complexity of the standards they thought their customers would require them to support. This appeared to irritate a lot of people, especially because HP had previously admitted that POSIX had generated a very large dollar volume of sales for them. This was exacerbated when the HP person making this presentation made sure he was considered an invited guest because HP did not want to appear to be supporting PASC by paying the meeting fees.

However, there was some valuable content underneath the rhetoric and wounded feelings. The name POSIX has considerable value in the marketplace. It was HP's contention that this value was in the core standards, but that most of the current work was aimed at extending the standards into specific niches (especially realtime and Ada), which are eroding the name recognition value of the core standards.

After a couple hours of debate, there was some acknowledgment of this position, so some ad hoc groups were formed to make suggestions on how we should proceed. I think some good progress was made at this meeting, but it was tainted by the fact that HP refused to participate in working toward a solution. Some reference was made to lobbing in a grenade, then running.

There are many possible outcomes before this issue is resolved, but I think one of the most likely will be some changes in terminology. The core standards will be given one name (like "Core POSIX" or "POSIX Classic"), and most of the new work will be given a different name. The biggest problem will be how to deal with the upcoming revisions of POSIX.1 and POSIX.2, which will undoubtedly be discussed in great detail at the October PASC meeting in Reno, NV.

Another issue brought up was the standards work in areas that were considered by some to be outdated. Security was the most often used example. This is a much harder problem to solve in general because of the process rules under which we live. If people are willing to work on a standard, and the balloting process...
approves it, we have no choice but to process it into an approved standard.

The real concern here was that the wrong technology would be standardized: the old instead of the new. There is a simple solution to this particular problem, but it requires the active participation of industry. If people come forward wanting to standardize the new technology and are willing to put in the time and energy to produce it, it will become a standard. If not, it will not. It's that simple.

The more general changes in standards involve high-level policy and procedure changes being planned or made by the IEEE. At the July PASC meeting the IEEE Standards Board vice chair made the first public presentation of the new Standards Association that the IEEE is forming, which will begin operation in January 1998. There are many changes involved with the creation of the SA, but the most significant is that only SA members will be allowed to cast ballots on proposed IEEE standards.

The change is fairly minimal for individual members: the membership fee is only $10 per year if you are a member of IEEE or the Computer Society. Remember, you already have to be a member of one of these to vote on standards now.

The big change is that corporations or other organizations (such as government agencies, educational institutions, or consortia) may also become SA members (at a much higher cost, of course). The plan is that balloting on a draft standard may be done by either the current method of individual voting or by a new method of corporate voting. The method would be selected when the initial PAR (Project Authorization Request) is submitted. If the corporate model is selected, then each organizational entity would get only one vote, but it would be guaranteed that vote, even through the substitution of the original person assigned to ballot.

This is a major change that will take a while to sink in and to get all the wrinkles ironed out. If you have comments or concerns about these changes, please feel free to send me email at <ljg.johnson@ieee.org> [or to me, <nick@usenix.org> -- ed].

The Standards Board is also making some small changes in the current process that should reduce the effort (and stress) in bringing a standard through the balloting process. At its last meeting the board agreed in principle to allow people to be taken off a ballot group if they request it. This will solve several current problems caused by people not returning their ballots, the most common reason being they changed jobs, are no longer interested, or in some cases, actually died during the balloting process. This is a significant change because the composition of a balloting group has always been immutably fixed.

The standards department staff and several of the individual sponsors (including PASC) have defined fast track processes to speed the standards process for specifications that have already been developed in another forum. Unfortunately, we have not yet had a project to test these defined processes. PASC tried to get Sun to submit some of the Java specifications for fast track in IEEE, but they have chosen to try to go directly into ISO JTC1 with their specifications, so we are still looking for volunteers. If anyone knows of a project that may be suited to this process, please contact me.

Year 2000 Test Methods
Christina Drukala <christinad@abtcorp.com> reports on the July 1997 meeting in Nashua NH.

ABT Corporation, a project management software developer, was faced with creating an internal Test Methods document to test readiness for their applications for the millennium change. While researching samples of cases, functions, and features at risk, we found a void of information. However, there was a plethora of Strategic data defining the problem and describing the disaster we were all going to experience if we did not correct this problem in a timely fashion. The available information discussed the maturity of processes necessary to implement a correction program, including the disciplines to be involved and the cost of remediation. This is valuable data but did not enumerate the tasks a Test Engineering department should perform to validate an application system, etc. So we started to develop our own test methods.

This internal ABT Corporation document was reviewed by the IEEE Year 2000 Study Group formed at the Jackson Hole, Wyoming PASC meeting in April 1997. At that meeting, the group adopted the ABT guidelines as the starting point for creating a Year 2000 Test Methods Recommended Practice. The core document is being developed by a group of Test Engineering and Quality Assurance professionals, together with members ranging from government employees to lawyers.

The recommended practice is not intended to be a certification standard. Its intent is to serve as an outline for personnel responsible for the testing of applications that have a finite size for storing date information in the event that the size is not sufficiently large. The most obvious case of this is holding a year in a two-digit format, and handling the case of 99 rolling over to 00, but the practice
will be equally applicable to other counters (e.g., the 32 bit POSIX time_t value in 2038). The document helps identify those modules known to have problems; and this list can be compared against the functions and procedures used by the system, application, and firmware. A single module in the application may have similarities to several of the general modules listed in the document. In addition, the document provides sample test scenarios that include purpose, environment, steps for replicating, and expected results that map to the modules/functions in the risk section of the Recommended Practice document. A small section includes test methodology along with definitions of remediation techniques.

For the last two meetings, in April and July, the group has been meeting as a study group, but is now ready to submit the Project Authorization Request (PAR), and expects sponsorship to be recommended in the October PASC meeting in Reno, Nevada. Before that, in late August, a core group will meet in Reston, Virginia for a two-day formal review of the Recommended Practice draft. Ideally, this document will go to IEEE ballot in early Spring 1998, in time for a summer general availability. Obviously, with the subject in question, timeliness is key, and this will be an interesting challenge to the usual tardiness observed in the ballot process.

**The Open Group Distributed System Management**

Nick Stoughton <nick@usenix.org> reports on the Open Group Distributed System Management meeting in Copenhagen, July 28-30.

**Not So Fast!**

Well, in the last issue, I reported on the progress of XSMU in the Open Group Distributed System Management Specification Group. I thought that all was well, and the document would progress rapidly towards a Common Application Environment (CAE) specification. Alas, I was wrong. The Technical Managers Committee in the Open Group decided they were unhappy with the consensus achieved and the contention that remained over some issues and asked us to revisit the issue. This is happening as I write. There is a feeling among several of the members that system management command line utilities are not what is needed — they do not help sell hardware. Perhaps they are right. Perhaps system administrators should just learn different commands for every type of system they have to deal with. Perhaps portable administration shell scripts are a thing of the past. Perhaps we should all be using NT’s administration tools from now on, and the Open Group should simply worry about the “wider” issues, and to hell with the detail.

**Managing the IT Dialtone**

The following is extracted from a publication by the Open Group, describing their new vision:

What it means to be “open” in today’s world of IT is continually changing, not in the least because of the rapid evolution of the technology itself. The time is right to review the Open Group’s mission and strategy and adjust its key programs accordingly.

The new Internet technologies promise to change the way we live and work, from company intranets through electronic commerce to telemedicine. However, to enable business to extract the full potential of this technology, we need a global information infrastructure that is ubiquitous, trusted, reliable, and as easy to use as the telephone. The Open Group calls this the IT Dialtone, and they believe it can emerge only through an effective cooperation between suppliers and customers.

The ability to mix and match best-of-breed technology from multiple vendors to rapidly build solutions to the changing needs of business remains a key requirement. In addition, it is recognized that existing “legacy” systems will stay with us, and organizations need a way to protect their investments in these systems and the invaluable information they contain.

With the IT Dialtone as the vision, the Open Group intends to be instrumental in furthering these objectives, enabling businesses to get maximum value from information systems both within and beyond their organizations. As a trusted, neutral organization with (thanks to our members) the necessary in-depth technical know-how, the Open Group is ideally placed to lead in the realization of the IT Dialtone.

Our key task will be to cause the realization of the IT Dialtone – a viable, global, open information infrastructure. This will be supported by a trusted forum of customers and suppliers and underpinned by a range of cost-effective enabling services such as collaborative development, testing, and branding.
To do this, the Open Group must itself adapt by expanding its scope to address the whole challenge and by shifting its role from its former technology focus to become a true forum for information, discussion, and resolution. This involves increasing the influence of the “buy-side,” reducing financial reliance on a small number of large sponsors and aligning its technical programs more closely with the customer and supplier needs that the forum identifies.

Essentially, the concept is to provide a coherent set of services and facilities to provide a pervasive, ubiquitous, global IT infrastructure as easy to use as the telephone. The System Management Working Group is looking at what services are required to manage this infrastructure: what exists and is implemented, what is under development that needs to be steered into this endeavor, and what is missing and needs development work from the vendors. The IT Dialtone will be implemented in three phases, in as short a time frame as possible. Phase one gathers those identified, existing, implemented technologies and places them into the framework, possibly noting rough edges that will need future attention. This phase is under way now and needs to be completed by the end of this year. Phase two provides the technical infrastructure, with a far greater emphasis on reliability and security. Phase three is the long-term goal and should be largely in place by the end of 1999. The IT Dialtone is a significant shift in the Open Group’s direction.

No longer are source code portability and command line interfaces the primary focus for their specifications. Overall, end-to-end services are the new focus. Branding, long a cornerstone of the Open Group process, could now be applied to service providers, in particular, to ISPs. Managing the IT Dialtone means identifying interfaces and services required to build a management server, a manageable application, and a manageable platform. It also places demands upon network service providers for transporting, managing, and monitoring facilities required for that infrastructure.

It is early yet. But watch this space – the entire organization is now focusing on this almost exclusively.
the bookworm

Books reviewed in this column:

Donald A. Knuth
The Art of Computer Programming

Robert Englemaner
Developing Java Beans

Peter van der Linden
Not Just Java

David Flanagan
Java in a Nutshell

Ralph E. Griswold & Madge T. Griswold
The Icon Programming Language

Bjarne Stroustrup
The C++ Programming Language

Brent B. Welch
Practical Programming in Tcl and Tk

Mark Harrison et al.
Tcl/Tk Tools

Mark Harrison & Michael McLennan
Effective Tcl/Tk Programming

Arnold Robbins
Effective AWK Programming

by Peter H. Salus

I've received an abnormally large number of really good books over the past few months— even three on Java that made me relent a bit. Perl, Tcl, C++, Icon, Awk, intranets, extranets, Linux, and security are among the other topics.

But the most exciting book to reach me was the new, third edition of Knuth's *Art of Computer Programming*, vol. 1. That's "Fundamental Algorithms." Knuth is redoing the other volumes as well. I expect to do an entire column on the "set" when it's complete. This is a great and important work, and one that most of you probably are familiar with. It looks as though Knuth has incorporated every erratum and thought about every comment made on the two first editions over nearly 30 years. The results are just stunning. I'm not certain you can be a real programmer and not have read at least part of Knuth (I admit to skipping).

Java
I guess I may as well get it over with.

Engelander's *Developing Java Beans* is an excellent introduction to component architecture and the use of Java Beans in programming. Because Java Beans can be used in graphical programming environments and can be used like "Lego blocks," Engelander (and others) claim that applications can be constructed with the writing of any code. I feel I've heard that before. But this is a solid and worthwhile book.

*Not Just Java* purports to be for "everybody who needs to understand the Internet revolution." I'm not sure that anyone truly understands that phenomenon, but this book does explain Java, CORBA, IOP, ActiveX, and a lot of other things. Unfortunately, the book is overwhelmingly pro-Sun in a nagging, chauvinistic manner that wore on me. It also eschews references and bibliography. There is an index reference to Scott McNeely, but not to James Gosling.

Flanagan has done an outstanding job on the second edition of *Java in a Nutshell*. Java, as you know, is a moving target, but this edition covers Java 1.1. It's both a reference and a solid tutorial on "inner classes." The 40+ pages on Java Beans are excellent.

Languages
Most of these books aren't new; they are all extensively revised and expanded editions of reliable favorites.

Icon has long been one of my favorite languages. I was a SNOBOL user in the 1960s, and Ralph Griswold was one of the creators of that while he was at Bell Labs. Mike O'Dell and I published an Icon paper in *Computing Systems* (2:4 [1989]). My 1983 copy of *The Icon Programming Language* is held together by a rubber band. I was thrilled when I got my copy of this third edition. Icon is a high-level general purpose language. Thanks to Ralph and Madge for taking the pains that this superb volume must have taken.

Stroustrup's third edition of his C++ tome has waxed: the first edition had 328 pages, the second, 669 pages; it is now 910 pages. Unlike those annoying volumes whose pages are filled with screen dumps, this is full of real information. Bjarne has done a splendid job in this total rewrite of his important work.
The second edition of Welch's Tcl/Tk book covers the Tcl 8.0 and Tk 8.0 releases and covers network sockets and using Tcl for Internet scripting. There's a CD-ROM with Tcl and Tk versions for Windows95, Windows3.1, WindowsNT, Macintosh, and UNIX. As John Ousterhout mentioned at the Tcl/Tk Workshop, the promotion of Tcl is a political thing at Sun. Welch's book may help it break out from being such a well-kept secret. Right now it might be the best language for Internet scripting.

For a long time, Ousterhout's and Welch's works were all there was (well there are others, but I find it hard to take Visual Tcl or Tcl for Dummies at all seriously) and Libes's volume is confined to expect). Now two more books have entered the lists. My advice is to get both.

One is Tcl/Tk Tools. Sixteen contributors, including Ousterhout, McLennan, and Libes, combined with Mark Harrison to produce this extraordinarily useful volume. But be warned: this is no book for beginners. Though it is clearly written and lucidly presented, it is no quick or easy read. But the results of careful reading will be very profitable to you.

At the Tcl/Tk Workshop in Boston, Addison-Wesley gave away two chapters of Harrison and McLennan's forthcoming book. Because it will be out by the time you read this, I'd like to say a few words. One of the chapters is on the canvas widget, which Brian Kernighan has said he considers one (if not the) most valuable. Harrison and McLennan have devoted 70 pages to canvas. If you're using Tcl at all, this book is a must.

The second edition of Robbins's Effective AWK Programming isn't a thorough redoing of the first edition. It's a corrected edition, updated for GAWK 3.0.3 and with a reference card bound in. The card is very useful. Oz Yigit hated the typography of the previous edition. That hasn't improved, but I'm not sure it matters: if you're not using GNU Awk, this is a very useful book.

The second edition of Learning Perl has acquired Tom Christiansen as Randall Schwartz's co-author. The llama has gained weight, but not as much as might have been anticipated. It's definitely the best introductory book on Perl.

Linux

Linux books are sprouting as though they were Java. Most of the ones I've seen are less than wonderful. However, Sobell's Practical Guide is truly first-rate. After introducing the system and the filesystem, Sobell proceeds to shells and (wonder of wonders!) actually discusses ssh intelligently. He also does a fine job on the utility programs.

Shells

Arthur and Burns's fourth edition still confines itself to the Bourne, C, Korn, and BASH shells. I wish someone would do a book on zsh. In fact, I wish someone would do a book on all the UNIX shells. This is the fourth book on shells in under a year. And they all cover the same stuff. Arthur and Burn's is above average, but not the best.

Nets

Extranets informs me that "Extranets, the next generation of Intranet, are dynamic wide-area networks that link a company's employees, suppliers, customers, and other key business partners in a secure, electronic on-line environment for business communications." That's overblown enough that I nearly threw the book away. I'm glad I didn't. Although I found Extranets verbose and occasionally unfocused, it's far from worthless.

Bort and Felix's Building an Extranet is both shorter and has a really good chapter on vendors/sources. Unfortunately, other books aren't resources - there are no references and no bibliography.

Hinrichs's Intranet is sort of book Dilbert's manager would love: it will give him all the jargon he needs with no need
to actually understand anything. It contains an “executive briefing” and “critical lessons.” It has a section on how “networks evolve to the Intranet.” Wouldn’t Darwin be pleased!

Web Security & Commerce is a first-rate book; it not only supplies the reader with a lot of well-presented information; it contains a list of online and print resources. It is more than a slimmer version of the authors’ 1,000-page epic on UNIX security; it concerns non-UNIX systems, and talks about encryption, SSL, digital signatures, etc.

Smith’s Internet Cryptography provides an excellent overview of the strengths, weaknesses, and uses of cryptography. But if what you need is a how-to, the second edition of Bruce Schneier’s 1996 book is the one for you.

Coda
Remember, next issue is December and will contain my traditional top ten list. If you fear I might miss something, email me.

Aviel Rubin, Dan Geer, and Marcus Ranum

Web Security Sourcebook

Simson Garfinkel and Gene Spafford

Web Security and Commerce

Reviewed by Rik Farrow
<rik@spirit.com>

I was just finishing up a report for a client who wanted to expose a Web server to the Internet and provide controlled access to some confidential data, but I delayed my final report until I could check these new books. I was concerned that I might have overlooked something or missed some new ideas that would make their Web server safer.

The first book I received was the Web Security Sourcebook. As I know two of the three authors, I really looked forward to reading it. Also, I had specific requirements relating to Web server security, so I started near the back, looking particularly for information about where they would put the Web server in relation to the firewall, how SSL works, and other Web server configuration tips.

Generally, this book is well written. I especially enjoyed reading the sections that compared and contrasted methods for making secure payments, which were clear and show a good understanding of the major players and protocols. Keep in mind that whoever controls electronic exchange stands to make money any time digital cash changes hands. This is big business.

I was less satisfied with specific information about SSL and Web server configuration. As for location of an exposed server with regards to the firewall, the book gently tends to suggest using a third network, a DMZ, connected to the firewall itself. For configuration of the Web server, things get confused. In one paragraph (page 138), you are advised to create a special user account and make its home directory the document root. In the very next paragraph, the same account is supposed to use the server root for its home directory. The instructions do suggest making root the owner of the server root instead of the special user account (an excellent idea).

Other small inconsistencies marred this book for me. In the section about adding a password to Netscape Navigator, the authors say, “the password does not appear to be stored on disk.” This annoyed me — where else would a persistent password be stored? A little experimentation found that changing the Navigator password caused the file key.db to be modified. The nobody user account (-2 in the book) is described as an “otherwise illegal UID.” The length of time given to crack a 40-bit RC4 key was 18 hours, but this was done in 3.5 hours in January of 1997 [<http://www.rsa.com/realabs/97challenge/>].

Web Security and Commerce covers pretty much the same territory but in more detail. Both books talk about browser security (just Internet Explorer and Navigator), active content, digital certificates, cryptography, Web server security, CGI scripts, and on-line payments. The difference is in the level of detail. I had wanted to know more about SSL. Web Security and Commerce includes a chapter on SSL, with an appendix detailing the actual protocol. Web Security and Commerce explains that SSL uses port 433 and that to use SSL in a form, you must specify https as the protocol type in the URL.

The Garfinkel and Spafford book also takes more time to deal with active content. Coverage of Java is fairly similar in both books, with many of the same papers being cited and explained in each book. Web Security and Commerce spent an entire chapter on ActiveX, while the
Rubin et. al. book has a half a page. Authenticode gets another half page there.

Both books talk in detail about digital certificates. Server certificates are used with SSL (for encrypting the content of forms, and the results of a query when SSL v3 is used), and the reasons for having a certificate signed by a Certificate Authority are clearly explained. Both books explain how to view site certificates when they have been cached by your browser (your browser will have at least the certificates for some certification authorities or CAs).

Web Security and Commerce recommends putting the Web server on the firewall’s DMZ. I agree. Recommendations for administering this exposed Web server would have been nice to find in either book. The Web Security Sourcebook does provide some details about creating a bastion host and suggests using encrypted links for remote administration. I had hoped to find suggestions on how to set up and run mirrored Web servers. In this approach, your Web administrators manage an internal Web server, and software automatically and securely mirrors it to the external Web server. Bill Cheswick had written stage to do this work at <http://cm.bell-labs.com/who/ches/>.

You can find an encrypting file transfer tool on Marcus Ranum’s Web site <www.clark.net/pub/mjr>. I have recently learned of rsync <ftp://samba.anu.edu.au/pub/rsync>, which can use SSH as an encrypted link. Neither book mentions any of these mirroring techniques.

Having said all that, you might be surprised to hear that I liked both books. The Rubin, Geer, and Ranum book simply has less detail than the Garfinkel and Spafford book. There are times when Garfinkel’s writing can be a bit annoying, like when he describes a PGP key signing party: “... whip out their private keys, and then have an orgy of public key encryptions as their private keys are pressed against each other.” I suggest that you get both books; they do make good reading and provide good resources, regardless of my nitpicking. But if your budget supports buying only one book, Web Security and Commerce would be the better choice.

Gary McGraw and Edward W. Felten

Java Security: Hostile Applets, Holes, and Antidotes


Reviewed by George W. Leach
<gwleach@gle.net>

Are you concerned about the pedigree of that Java applet running in your Web browser? Perhaps you are considering building your own Java application for use on your company’s intranet or extranet (sorry for the buzzwords)? You’ve heard that Java has been designed with security in mind, but you need to know more. Do you understand the security implications of adding directories to the classpath variable? Do you understand Java’s security mechanisms? This is the book to answer your questions and concerns.

The authors are Gary McGraw of Reliable Software Technologies and Edward Felten of Princeton University’s Safe Internet Programming (SIP) team. SIP is the team at Princeton that has been uncovering security problems with Java, ActiveX, JavaScript, and other Internet technologies since the emergence of these technologies in the fall of 1995. The list of people who reviewed the book in its formative stages reads like a Who’s Who of the Java security field.

The first couple of chapters provide a high-level overview of Java and the security concerns that introducing it into your computing environment bring. The Java security model is explored as well. The authors then concentrate on exploring various holes in the Java security model that have been discovered and fixed over the past couple of years to illustrate the types of problems that can be encountered with Java. These types of problems are lumped under the category of attack applets, which exploit bugs in the security model to compromise the machine.

Another category of Java security concerns is known as malicious applets. These types of applets include denial of service attacks, spoofing email, stealing CPU cycles, and other annoying Java tricks.

The final two chapters of this book discuss steps that can be taken by Web surfers to avoid or minimize security risks associated with Java applets and future security features in Java, many of which have appeared in version 1.1.

A couple of appendices provide the Java Security Frequently Asked Questions (FAQ) as of July of 1996 and the relevant Computer Emergency Response Team (CERT) Alerts pertaining to security problems with Java discovered during 1996. Both of these appendices are way out of date, but the authors point to online versions of these appendices at <http://www.cs.princeton.edu/sip/java-faq.html>.

Although none of the information presented in this book is new, it is all consolidated into a well-written, concise volume that can be read with ease in several hours even by novices to Java. Another benefit of this book is an objective viewpoint of Java security that strives to stay clear of the marketing hype that surrounds the language.

One of the problems with this book or any book written about any aspect of the Internet is the currency of the information. Fortunately, the book was written with this in mind. Since its publication, the authors and their colleagues have discovered additional flaws in Java, which are discussed online at <http://www.cs.princeton.edu/sip/News.html>.
Web Client Programming with Perl


Reviewed by Carolyn Sienkiewicz
<csienki@pop500.gsfc.nasa.gov>

“Web client programming? Why would you do that?” This was the first impression I experienced of this book, because these were the first words uttered by an esteemed colleague (a software developer) upon seeing me (a system administrator) looking at the cover of this book. I indicated a phrase at the top of the book’s cover and replied, “Well, apparently you might want to automate tasks on the Web.”

Web client programming wasn’t an activity that had ever occurred to either my colleague or me. Neither of us is infatuated with the Web, and neither of us views the Web as much more than a resource for information or a potential application interface in our daily computer work.

First, the good news. This book explores automating Web tasks using Perl and the Tk extension to Perl. One example of a Web task that you might want to automate could be checking the validity of the links in a Web site that you maintain. I found this to be an entertaining thought to play with, and you might, too.

The author starts out by explaining what goes on behind the scenes when you use a Web browser. This material outlines the conversation that occurs between the Web server and the browser. Next he explains the structure of an HTTP transaction, along with the various client request methods, HTTP headers, and server response codes. All of this material is extremely elementary.

The meat of the book comes in the remaining chapters, which cover use of the socket library for Perl, the LWP library (the library modules for WWW access in Perl), and then plenty of example code in the form of client programs (primarily LWP based, and then graphical examples with Perl/Tk).

If you are a sophisticated, full-time Web administrator coming from a system administrator background and have done any Web client programming, then this book is going to be way too basic for you. The example programs do very simple things: check on your FedEx package delivery, check if servers are up, create a dictionary client. If you haven’t had the opportunity or need to do Web client programming, you’d probably have fun with this and may wish to check it out.

An audience that could potentially gain from this book would be nontechnical or neophyte Web administrators. Other audiences best suited to benefit from this work would include, to quote the author, “computer enthusiasts, tinkerers, and motivated students.”

Now for the bad news. Having tried to point out the positive attributes of the book, I really would not encourage login: readers to get it. The primary reason is that the author tends to gloss over and simplify things to the point that his explanations lose clarity and precision and can be misleading or confusing. As an example, when he dives into what he always refers to as “the socket library” in chapter 3, he never points out that this is a socket package that is a part of Perl. I immediately thought, “Is he saying that this is a system library?” After all, at this point in the text, no Perl code has appeared yet, and no real mention of Perl has occurred. Additionally, the socket calls bear a strong resemblance to, but are not identical to, the system socket calls. I kept wanting to write “Perl” in front of every instance of the phrase “socket library” to make the language more precise. This might seem like a petty example, but unfortunately, this type of glossing over of details is so prevalent throughout the book to be annoying.

I think I can understand authors wanting to leave some level of detail out of their books. This is often done in an attempt to appeal to a broader audience. However, in this case, I think the omissions can be harmful. After all, the book does state the author expects you to have a solid knowledge of Perl. If you know Perl, I am quite sure you can handle much more precision and accuracy of detail than this work provides. Unfortunately, those who may be less knowledgeable are likely to ingest imprecise information, gain imprecise or erroneous understanding, and then have to unlearn it later.
John Zukowski

Java AWT Reference
Pp. 1045. $39.95.

Reviewed by Bruce O'Neel
<beoneel@macconnect.com>

That groaning you hear from your book store is just the foundations trying to support yet another shipment of Java books. One of the better new books is John Zukowski's Java AWT Reference. He has written a huge book on that large and sometimes confusing array of classes known as the Java Abstract Window Toolkit (AWT). AWT provides the graphical user interface for Java programs, and its complexity has produced a plethora of books. This book covers both the original 1.0.2 version as well as the new improved 1.1 AWT. One of the reasons why the book is so large is that version 1.1 has a lot of changes from 1.0.2. The other reason the book is large is that AWT is just a large class library.

The book is divided into 23 chapters and 4 appendices. The first 17 chapters are the first half of the book, and they are a guide to using AWT, both 1.0.2 and 1.1. The last 6 chapters are a class reference organized alphabetically by package. Rather than include a CD-ROM, the author has an FTP site that allows you to get the sample code. I found this nice because I didn't feel that I needed yet another copy of the JDK on CD-ROM.

This should not be your first Java book. Rather than try to teach everything in one book, this just covers the AWT. You must know Java fairly well before this book will make sense.

The guide section has an overview and then covers the following Java packages: java.awt, java.awt.image, java.awt.event, java.awt.datatransfer, java.awt.peer, and java.applet. A nice feature of this section is that the chapters talk about user interface concepts such as containers and layouts and input fields that you would find easy to reference when programming. Are you doing an input text box? You could see from the table of contents that you want chapter 8 on input fields. Once there, you get a nice explanation of how to create and use TextComponents, TextField, and TextArea, along with some nice examples. You also would be able to see which parts of these classes you could use if you were running under 1.1 vs. 1.0.2.

I found the chapter on events and the chapter on layouts to be the two most useful chapters. The largest change from a design pattern point of view between 1.0.2 and 1.1 is in event handling. A short explanation is that in 1.0.2 when an event such as a mouse click is generated, the AWT finds a component that might be interested in this event, and then that event handler is called with that event, and the component has the option of processing it. If that event handler chooses not to process it, then AWT goes off to find another component that might be interested. In 1.1 your program objects that are interested in a particular event register for that event and are called when that event occurs. For a better explanation, see chapter 4.

The layouts chapter did a good job of explaining all five different LayoutManagers. LayoutManagers decide where to place components within containers and allow your program to run independent of the screensize and resolution. LayoutManagers also allow your windows to be resized without you having to do a lot of work to redisplay your components. The explanation of the GridBagLayout was quite clear and, if we're lucky, should cut down on the questions in <comp.lang.java.*>.

The reference section of the book has one chapter for each of the Java packages covered. This gives you a quick reference when you know what class you want but forgot what the member function is called. This is also nice for those of us who like to flip pages rather than wait for some online help system to bring up documentation on our too-small screen. It would have been nice to have a reference either to page number or chapter number where you could find the explanation and/or an example of a class in the reference section. Even so, most of the time it is very obvious from the table of contents which chapter to go to for more explanation.

In the jumble of Java books, is this a useful one? I felt it was. I enjoyed reading the guide section, and it is the book that sits next to my Powerbook when I'm programming. Is it the best AWT book? I'm not sure that is an answerable question because reading all the Java books is at best a Sisyphean task.

October 1997 ilogin: 75
As the Internet continues to grow and expand, people have tried to define, model, and understand it. In *Internet Economics*, the economists take their turn. This book developed out of a conference held at MIT in 1995. The focus of the articles is the Internet architectural models and their implication for pricing and pricing policies in general. The stated goal of the book is to develop a metric for economic analyses of Internet transactions. I’ll talk more about that later.

There are six sections to the book: Introduction to Internet Economics, The Economics of the Internet, Interconnection and Multicast Economics, Usage-Sensitive Pricing, Internet Commerce, and Internet Economics and Policy. The number of articles in each section varies. The articles also vary in how much economic theory they contain. Only one or two have detailed derivation of formulas and equations, so most of the material is accessible to those without an economics background.

The technical detail doesn’t require a degree in computer science. There is a brief description of protocols, enough to understand the development of the architecture models. Because economics involves optimizing resource allocation, it is important to understand the difference between some of the protocols. But the discussion doesn’t get into the technical issues, such as increasing the size of the Internet address space.

Trying to find a good model of the Internet has some potential benefits. Presumably, the economists could use it to devise a better scheme of charging for Internet services, which presumably would mean users would pay for the resources they use. Unfortunately, most of the payment schemes mentioned in this book involve usage-based charges. It makes sense if you are trying to distribute the charges to those who actually used the service. The book generally discusses two options for charging for services: the usage-based system I just mentioned and a flat fee scheme that is what most current ISPs use. The problems with the flat fee system, from the perspective of developing a model, are that everyone pays for the infrastructure, whether they use it or not, and the difficulty of determining exactly what part of the infrastructure is justifiably included in the flat fee. Because this system doesn’t optimize the charges, it is usually rejected by the authors or is used as the example of what we don’t need. Interestingly, the book doesn’t mention the market forces that would solve the two problems. Basically, the ISPs will continually adjust their prices until they can pay for everything and still increase their customer base. That is what has been happening in the two years since the conference was held.

Most everyone agrees that usage-based charges would be better. Most of the economists writing for this book feel that they would prefer a usage-based system and have some good arguments in support of that type of system. Most users of Internet services would probably prefer that. Personally, I wonder about the flat fee because I don’t do a lot of bandwidth-intensive work through my ISP and occasionally have troubles calling into my ISP. Why should I pay for the time that someone else is downloading lots of graphics?

There are some serious problems with usage-based schemes, from both a technical and an economic perspective. Economically, the problems involve defining exactly what a resource is. Of all the equipment and skills required to maintain an ISP, exactly how much of each should be allocated to the different Internet services provided? This also involves technical issues. Many Internet services maintain a continuous connection, such as Telnet. Others have intermittent connections, at least from the ISP to the host site, such as the HTML World Wide Web protocol. Both services require a continuous connection from the user’s computer (or terminal) to the ISP. Telnet also requires a continuous connection from the ISP to the host, and HTML will have numerous short connections. Even worse, the bandwidth required for Telnet is much less than that required for graphics images downloaded to a browser. It gets even more complicated when you factor in network infrastructure and new services.

There is a major technical issue that is described as part of the problem with implementing a usage-based system. Even if you decide on what resources to monitor, how do you monitor them? The necessary hardware and software would have to be designed, built, and installed. This would add to the expense of operating an ISP. In addition, this monitoring would be using resources itself. Plus it is not clear that all resources would be something that was easy to monitor.

The Internet has refused to be categorized or defined before. Apparently, it will add economists to the list of those who have discovered this difficulty. Despite the efforts of those who contributed to this book, they cannot resolve the difficulties of defining the terms to switch to usage-based pricing for Internet services. The one theme through the book is that although usage-based pricing is preferred, it is too difficult to define with our current understanding of the resources required. Until those issues get resolved, a flat fee system is preferred because it is easier to implement.

This is an interesting book. That it can be the subject of such detailed analysis is probably a sign that the Internet has “arrived.” As a technical professional, I
never looked at some aspects of an Internet connection in this way before. It certainly opened my eyes as to what an Internet connection may actually cost – beyond just calling up ISPs and asking for quotes. As new services are proposed, these issues may become even more important. With its continued growth, the Internet is reaching a point where some resources are saturated (some will even claim they are oversaturated).

Usage-based charging may be necessary to relieve some of the saturation so it is important to begin discussing these issues. As a first step in that direction, I recommend reading this book.

Bryan Costales with Eric Allman

sendmail, 2d ed.


Reviewed by Steve Hanson

<shanson@htc.honeywell.com>

Many of the O’Reilly system administration books have been published in multiple editions. Normally, the new editions of these books provide minor updates to keep sysadmins up to date on the latest technologies. The second edition of sendmail is a major rewrite, and the new edition is a necessary addition to the bookshelves of any system administrator who deals with sendmail-based email systems. The original version of sendmail (the bat book) quickly became the standard reference for understanding the intricacies of this often confusing program, and three years later the second edition provides the latest information for the current sendmail 8.8 releases.

Some information has been dropped from the new edition, notably the sections detailing the configuration of versions of sendmail previous to version 8 and other offshoots such as IDA and K/S sendmail. Most manufacturers have updated their sendmail systems to version 8, and if you are blessed with a system running an older version, you’re probably better off updating than understanding how to configure an old release.

What is new in this edition is much more detailed information on using the m4 configuration system in sendmail 8. Use of the m4 configurations allows admins to build the complex sendmail.cf file from simple m4 files, which are much easier to understand. Personally, I haven’t written a sendmail configuration from scratch since I started using IDA sendmail and its m4 system quite a few years ago. Almost any sendmail configuration can be built using only the m4 configurations. The new configuration information is covered throughout the second edition while different sendmail concepts are discussed. This is a big improvement over the last edition, which spent many chapters describing the details of sendmail.cf rules while relegating the information on m4 files to an appendix.

The second edition also provides detailed information on a number of other topics that were given short shrift in the first edition. The newer book carefully explains the use of the -bt options for sendmail rule testing. It also provides information on the often bewildering enhancements that Sun has made in its own versions of sendmail, including the recent Sun sendmail releases that will support both the old (version 5) sendmail config rules that Sun had supported in the past, and the new modified 8.7 sendmail configuration rules that are allowed in Solaris 2.5 and later.

Additional coverage is given to the use of Berkeley DB databases in sendmail configurations as well.

The final, and perhaps most timely, improvement in the second edition is the inclusion of much more information on the checkcompat routine in sendmail. Although checkcompat has existed in sendmail since version 3, the utility of the routine has been unclear in most documentation. In the last few months the amount of spam mail on the Internet has increased at an amazing rate, and an understanding of the creative use of checkcompat will allow your site to filter out email from offending sites, among other uses. So if you can’t handle one more irate user coming into your office begging you to fend off the latest get-rich-quick scheme to float in over the wire, read the new checkcompat chapter in this book.

In general, this update to the much beloved bat book cleans out some of the outdated older information in the first edition while expanding on the information presented for sendmail 8, and adding even more information on the latest version 8.8 releases. Even if you have the original sendmail book, the new updated version is worth purchasing.
**TUTORIALS**
42 tutorials covering: tools, security, Web, intranets, networking, sendmail, software development, Perl, legal issues, and professional development.

**TECHNICAL PROGRAM**

<table>
<thead>
<tr>
<th>Wednesday, Oct. 29</th>
<th>Refereed Papers Track</th>
<th>Invited Talks Track</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00am–10:30am</td>
<td>Opening Remarks</td>
<td>Overview of the Large Scale System Administration of Windows NT Workshop</td>
</tr>
<tr>
<td></td>
<td>Hal Pomerantz, Deer Run Associates; and Celeste Stokely, Stokely Consulting</td>
<td>The program chairs of USENIX's Large Scale System Administration of Windows NT Workshop will select highlights of the workshop for presentation.</td>
</tr>
<tr>
<td></td>
<td>Keynote Address: Generation X in IT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Randy Johnson and Harris Kern, R &amp; H Associates, Inc.</td>
<td></td>
</tr>
</tbody>
</table>

**11:00am–12:30pm**

<table>
<thead>
<tr>
<th>Monitoring</th>
<th>Monitoring Application Use With License Server Logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Chair: Adam Moskowitz, Genome Therapeutics Corporation</td>
<td>Jon Finke, RPI</td>
</tr>
<tr>
<td>Implementing a Generalized Tool for Network Monitoring Marcus Rasmus, Network Flight Recorder, Inc.</td>
<td></td>
</tr>
<tr>
<td>Extensible, Scalable Monitoring for Clusters of Computers Eric Anderson and Dave Patterson, University of California, Berkeley</td>
<td></td>
</tr>
<tr>
<td>Overview of the Large Scale System Administration of Windows NT Workshop</td>
<td></td>
</tr>
</tbody>
</table>

**2:00pm–3:30pm**

<table>
<thead>
<tr>
<th>The Business of System Administration</th>
<th>DNS—Doing Nothing the Same</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Chair: Wendy Nather, Swiss Bank Warburg</td>
<td>Joel Avery and Andrew Macpherson, Nortel Technology</td>
</tr>
<tr>
<td>Automating 24x7 Support Response To Telephone Requests Peter Scott, NASA Jet Propulsion Laboratory</td>
<td></td>
</tr>
<tr>
<td>Turning the Corner: Upgrading Yourself from System Clerk to System Advocate Tom Limoncelli, Lucent Bell Labs</td>
<td></td>
</tr>
<tr>
<td>How To Control and Manage Change in a Commercial Data Center Without Losing Your Mind Sally J. Howden and Frank B. Northrup, Distributed Computing Consultants, Inc.</td>
<td></td>
</tr>
</tbody>
</table>

**4:00pm–5:30pm**

<table>
<thead>
<tr>
<th>System Design Perspectives</th>
<th>Panel Discussion: 40 Bosses, 3000 Users, 20 Projects... Managing Computers for Academics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Chair: Hal Pomerantz, Deer Run Associates</td>
<td>Chair: David Parter, University of Wisconsin</td>
</tr>
<tr>
<td>Developing Interim Systems Jennifer Caetta, NASA Jet Propulsion Laboratory</td>
<td>Panelists TBA</td>
</tr>
<tr>
<td>A Large-Scale Data Warehouse Application Case Study Dan Pollack, America Online</td>
<td></td>
</tr>
<tr>
<td>SHUSE at Two Henry Spencer, SP Systems</td>
<td></td>
</tr>
</tbody>
</table>

**Thursday, October 30**

<table>
<thead>
<tr>
<th>Working with PCs</th>
<th>Logging and Monitoring: How, Why, and When</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Chair: Melissa Binde, Swarthmore College</td>
<td>Peter Honeymen, CITI, University of Michigan, and Joe Saul, ITD DPD&amp;E, University of Michigan</td>
</tr>
<tr>
<td>A Web-Based Backup/Restore Method for Intel-Based PCs Tyler Barnett, Lexmark International</td>
<td></td>
</tr>
<tr>
<td>Managing PC Operating Systems With A Revision Control System Gottfried Rudorfer, Vienna University of Economics and Business Administration</td>
<td></td>
</tr>
<tr>
<td>BAL: A Tool to Synchronize Document Collections Between Computers Juergen Christoffel, GMD</td>
<td></td>
</tr>
</tbody>
</table>

**VISIT OUR WEB SITE:** http://www.useunix.org/events/lisa97
<table>
<thead>
<tr>
<th>Time</th>
<th>Inside the Black Box</th>
<th>Invited Talks Track</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:00am–12:30pm</td>
<td>Inside the Black Box</td>
<td>So Now You Are the Project Manager</td>
</tr>
<tr>
<td></td>
<td>Session Chair: John Sellens, UUNET Canada</td>
<td>William E. Howell, Glaxo Wellcome, Inc.</td>
</tr>
<tr>
<td></td>
<td>Increased Server Availability Through Failover Capability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Michael R. Barber, Michigan Technological University</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INN and the Cyclic News Filesystem</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scott Fritchie, Minnesota Regional Network</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adaptive Locks for Frequently Scheduled Tasks With Unpredictable Runtimes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mark Burgess and Demosthenes Skiperakis, Oslo College</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Work-In-Progress Reports (WIPs)</th>
<th>Invited Talks Track</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:00pm–3:30pm</td>
<td>Short, pithy, and fun. Work-In-Progress Reports introduce interesting new or ongoing work. If you have work you would like to share or a cool idea that is not quite ready to be published, a WIP is for you! We are particularly interested in presenting student work. To reserve your presentation slot, contact the WIPs coordinator via email to <a href="mailto:lisa97@usenix.org">lisa97@usenix.org</a>. A list of topics is announced on-site.</td>
<td>When UNIX Met Air Traffic Control</td>
</tr>
<tr>
<td></td>
<td>By Jim Reid, RTFM Ltd.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Net Gains</th>
<th>Invited Talks Track</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:00pm–5:00pm</td>
<td>Net Gains</td>
<td>Enterprise Backup and Recovery—Do You Need a Commercial Utility? (4:00pm–5:30pm)</td>
</tr>
<tr>
<td>(4:00pm–5:30pm for Invited Talks)</td>
<td>Session Chair: Josh Simon, Parernet</td>
<td>W. Curtis Preston, Pencom Systems Administration</td>
</tr>
<tr>
<td></td>
<td>Creating the Network for Lucent Bell Labs Research</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tom Limoncelli, Thomas Reingold, Ravi Narayan, and Ralph Loura, Lucent Bell Labs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Instrumenting and Tuning a Very High Performance Web Server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Douglas L. Urner, BSDI</td>
<td></td>
</tr>
</tbody>
</table>

**Friday, October 31**

<table>
<thead>
<tr>
<th>Time</th>
<th>Config Management</th>
<th>A Technologist Looks at Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00am–10:30am</td>
<td>Config Management</td>
<td>Steve Johnson, Transmeta Corp.</td>
</tr>
<tr>
<td></td>
<td>Session Chair: Paul Anderson, University of Edinburgh</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Automation of Site Configuration Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jon Finke, RPI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>An Anarchists Guide to Heterogeneous Configuration Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alva Couch, Tufts University</td>
<td></td>
</tr>
<tr>
<td></td>
<td>An Analysis of UNIX System Configuration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Remy Evard, Argonne National Lab</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Mail</th>
<th>IPv6 Deployment on the 6bone</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:00am–12:30pm</td>
<td>Mail</td>
<td>Bob Fink, Lawrence Berkeley National Labs.</td>
</tr>
<tr>
<td></td>
<td>Session Chair: Bill LeFebvre, Group sys Consulting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tuning Sendmail for Large Mailing Lists</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rob Kolstad, BSDI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selectively Rejecting Spam Using Sendmail</td>
<td></td>
</tr>
<tr>
<td></td>
<td>check_Rulesets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Robert Harker, Harker Systems</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A Better Email Bouncer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rich Holland, Rockwell Collins</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Panel: Is System Administration a Dead-End Career?</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:00pm–3:30pm</td>
<td>Panel: Is System Administration a Dead-End Career?</td>
</tr>
<tr>
<td></td>
<td>Moderator: Celeste Stokely, Stokely Consulting</td>
</tr>
<tr>
<td></td>
<td>Panelists: Bill Howell, Glaxo Wellcome, Inc.; Wendy Nather, Swiss Bank Warburg; and Hal Pomeranz, Deer Run Associates</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>The LISA Quiz Show!</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:00pm–5:30pm</td>
<td>The LISA Quiz Show!</td>
</tr>
<tr>
<td></td>
<td>Hosted by Rob Kolstad, BSDI</td>
</tr>
</tbody>
</table>
**Wednesday-Friday, October 15-17, 1997 • Santa Barbara, California**

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
</table>
| **Wednesday** | **8:30am—9:30am**  
Keynote Address: The Promise of Domain-Specific Languages  
Paul Hudak, Yale University, Department of Computer Science |
| **10:00am—11:30am**  
Service Combinators for Web Computing  
Luca Cardelli, Digital Equipment Corporation and Rowan Davies, Carnegie-Mellon University  
A Domain-Specific Language for Video Device Drivers: From Design to Implementation  
Scott Thibault, Renaud Marlet, and Charles Consel, IRIS/INRIA—Université de Rennes 1  
Domain-Specific Languages for ad hoc Distributed Applications  
Matthew Fuchs, Walt Disney Imagineering |
| **1:00pm—2:30pm**  
Experience with a Domain-Specific Language for Form-Based Services  
David Atkins, Thomas Ball, Michael Benedikt, Glenn Bruns, Kenneth Cox, Peter Mataga, and Kenneth Rehor, Bell Laboratories, Lucent Technologies  
Experience with a Language for Writing Coherence Protocols  
Satish Chandra and James R. Larus, University of Wisconsin; Michael Dahlin, University of Texas; Bradley Richards, Vassar College; and Randolph Y. Wang and Thomas E. Anderson, University of California, Berkeley  
Lightweight Languages as Software Engineering Tools  
Dionysis Spinellis, University of the Aegean and V. Guruprasad, IBM T.J. Watson Research Center |
| **3:00pm—5:00pm**  
A Slicing-Based Approach for Locating Type Errors  
T. B. Dinesh, CWI and Frank Tip, IBM T. J. Watson Research Center  
Typed Common Intermediate Format  
Zhong Shao, Yale University  
Incorporating Application Semantics and Control into Compilation  
Dawson R. Engler, MIT Laboratory for Computer Science  
Code Composition as an Implementation Language for Compilers  
James M. Stichnoth and Thomas Gross, Carnegie Mellon University |
| **8:30pm—11:00pm**  
Birds-of-a-Feather Sessions |
| **Thursday** | **8:30am—9:30am**  
Invited Talk: Synchronous Languages—An Experience in Domain-Specific Language Design  
Gérard Berry, École des Mines de Paris, Centre de Mathématiques Appliquées; INRIA, Projet Meije |
| **10:00am—11:30am**  
BDL: A Language to Control the Behavior of Concurrent Objects  
Frédéric Bertrand and Michel Augeraud, Université de la Rochelle  
A Domain-Specific Language for Regular Sets of Strings and Trees  
Nils Klarlund, AT&T Labs Research and Michael I. Schwartzbach, University of Aarhus  
A Modular Monadic Action Semantics  
Keith Winsborough and John Hamer, University of Auckland |
| **1:00pm—2:30pm**  
SHIFT and SMART-AHS: A Language for Hybrid System Engineering Modeling and Simulation  
Marco Antoniotti and Aleks Güllü, University of California at Berkeley  
Design and Semantics of Quantum: A Language to Control Resource Consumption in Distributed Computing  
Luc Moreau, University of Southampton, and Christian Queinnec, Université de Paris 6, INRIA-Rocquencourt  
Architectural Domains: A Framework for Characterizing Architectural Description  
Nenad Medvidovic and David S. Rosenblum, University of California, Irvine |
| **3:00pm—4:30pm**  
The Zephyr Abstract Syntax Description Language  
Daniel C. Wang, Andrew W. Appel, Jeff L. Korn, and Chris S. Serra, Princeton University  
ASTLOG: A Language for Examining Abstract Syntax Trees  
Roger F. Crew, Microsoft Research  
KHEPERA: A System for Rapid Implementation of Domain-Specific Languages  
Rickard E. Faith, Lars S. Nyland, and Jan F. Frins, University of North Carolina at Chapel Hill |
| **5:00pm—6:00pm**  
Invited Talk: Intentional Programming—An Ecology for Abstractions  
Charles Simonyi, Chief Architect, Microsoft |
| **8:30pm—11:00pm**  
Birds-of-a-Feather Sessions |
| **Friday** | **8:30am—10:30am**  
DISTIL: A Transformation Library for Data Structures  
Yannis Smaragdakis and Don Botony, University of Texas at Austin  
Programming Language Support for Digitized Images or, The Monsters in the Closet  
Daniel E. Stevenson and Margaret M. Fleck, University of Iowa  
Modeling Interactive 3D and Multimedia Animation with an Embedded Language  
Conal Elliott, Microsoft Research  
A Special-Purpose Language for Picture-Drawing  
Samuel Kamin and David Hyatt, University of Illinois at Urbana-Champaign |
| **11:00am—Noon**  
Invited Talk: Aspect-Oriented Programming—Improved Support for Separation of Concerns in Design and Implementation  
Gregor Kiczales, Principal Scientist, Xerox Palo Alto Research Center |

**Visit Our Web Site:** [http://www.usenix.org/events/dsl97](http://www.usenix.org/events/dsl97)
All the Power of Perl. No Waiting.

Why search the Net for quality Perl programming aids and wait for time consuming downloads? The Perl Resource Kit is the most comprehensive collection of documentation and commercially enhanced software tools yet published for Perl programmers. The UNIX edition, the first in a series, is the definitive Perl distribution for webmasters, programmers, and system administrators. Software tools on the Kit's CD include:

- A Java/Perl back-end to the Perl compiler, written by Larry Wall, creator of Perl
- Snapshot of the freeware Perl archives on CPAN, with an install program and a web-aware interface for identifying more recent online CPAN tools

Perl Resource Kit—UNIX Edition
By Larry Wall, Clay Irving, Nate Patwardhan, Ellen Siever, David Futato & Brian Jepson
$49.95 (est.), Includes CD-ROM

The Kit also includes four tutorial and reference books that contain systematic documentation for the most important Perl extension modules, as well as documentation for the commercially enhanced and supported tools on the CD. The books in the Kit are not available elsewhere (or separately) and include:

Perl Module Programmer's Guide
by Clay Irving and Nate Patwardhan
Perl Module Reference Manual (two volumes)
compiled and edited by Ellen Siever and David Futato
Perl Utilities
by Brian Jepson

Pre-order the Perl Resource Kit at LISA USENIX booth #100 and get a 20% discount.

TCP/IP Network Administration, 2nd Edition
By Craig Hunt
2nd Edition 1/97 (est.)
575 pages (est.)
ISBN 1-56592-322-7
$39.95 (est.)

Managing IP Networks with Cisco Routers
By Scott M. Balfour
1st Edition 10/97 (est.)
250 pages (est.)
ISBN 1-56592-326-0
$29.95 (est.)

Windows NT in a Nutshell
By Eric Pearce
1st Edition 6/97
364 pages
ISBN 1-56592-251-4
$19.95

Tel/TK Tools
By Mark Harrison
1st Edition 9/97
678 pages
Includes CD-ROM
$49.95

Web Security & Commerce
By Simon Garfinkel with Gene Stafford
1st Edition 6/97
506 pages
ISBN 1-56592-269-7
$32.95

101 Morris Street, Sebastopol, CA 95472
707-829-0515 • FAX: 707-829-0104
Orders/inquiries: 800-998-9938
Weekdays 6AM to 5PM PST

For more information about these and other O'Reilly Perl products, visit www.oreilly.com
15% Discount on 15 Best-Selling Addison-Wesley Books

Go to participating stores during the month of October to take advantage of a 15% discount on 15 of our best-selling books.

For a complete list of titles and participating bookstores, please see our web site at:

http://www.awl.com/cseng/solutions97/
UNIX System Programming Using C++

UNIX System Administration Handbook, 2nd Edition

Oracle and UNIX Performance Tuning

The TIME-LAPSE EXPLAINED

HP-UX 10.x System Administration

Easy Linux/NT System/Compaq Joes

Simpuy AIX

UNIX Shells by Example & Shell Quickie

UNIX System Administration Handbook, 2nd Ed.

Windows NT and HP-UX System Administrators' Guide

Internetworking with TCP/IP Volume III:

Network Administering - Sockets and XTI, and Ed.

UNIX Network Programming, Volume 1:

COMING SOON

0-13-390021-2 $54.95

UNIX System Programming Using C++

0-13-331555-6 $48.00

20% OFF TO USENIX ASSOC. MEMBERS

PRACTICE HALL PTR & BOOKPOOL TEAM UP TO OFFER

REAL BOOKS BY REAL AUTHORS FOR REAL PROFESSIONALS

Members of USENIX will find timely and useful information in the journals published by Sage Science Press. As such, we are pleased to offer USENIX members a 20% introductory rate to any Sage Science Press journal.

**Computer Modeling and Simulation in Engineering**

**Editor:** S.N. Atluri, Georgia Institute of Technology, Atlanta

...publishes application-oriented papers that utilize computer modeling and simulation techniques to understand and resolve industrial problems or processes that are of immediate and contemporary interest.

Quarterly: February, May, August, November
1996 Individual Rate: $70
1996 Discounted USENIX Rate: $56


...publishes original papers covering the broad field of experimental mechanics, focusing on "solid mechanics," the study of material phenomena as well as the failure of materials and structures.

Quarterly: March, June, September, December
1996 Individual Rate: $72
1996 Discounted USENIX Rate: $57

**The International Journal of Supercomputer Applications and High Performance Computing**

**Editor:** Jack Dongarra, University of Tennessee - Knoxville

...provides a forum for original research papers and timely review articles on the use of supercomputers to solve complex modeling problems in a spectrum of disciplines.

Quarterly: February, May, August, November
1996 Individual Rate: $75
1996 Discounted USENIX Rate: $60

**The Journal of Biological Rhythms**

**Editor:** Fred Turek, Northwestern University

...publishes original, full-length reports of empirical investigations into all aspects of biological rhythmicity, particularly rhythms related to the major environmental cycles.

Bimonthly: February, April, June, August, October, December
1996 Individual Rate: $89
1996 USENIX Rate: $71

**Journal of Vibration and Control**

**Editor:** Ali H. Nayfeh, Virginia Polytechnic Institute and State University

...provides a forum for original research on vibration phenomena occurring in the mechanical, structural, aeronautical, electrical, materials, chemical, control, ocean and environmental fields.

Quarterly: February, May, August, November
1996 Individual Rate: $70
1996 USENIX Rate: $56

**Mathematics and Mechanics of Solids**

**Editor-in-Chief:** L.T. Wheeler, University of Houston

...fosters innovative research in solid mechanics and materials science through publication of original and self-contained research that elucidates the mechanical behavior of solids with particular emphasis on mathematical principals.

Quarterly: March, June, September, December
1996 Individual Rate: $70
1996 USENIX Rate: $56

A sample copy of any Sage Science Press journal is available upon request. To order a copy of any of these publications or to request a free sample copy please use one of the following numbers:

**Sage Customer Service:** 805-499-9774  **Sage FaxLine:** 805-499-0871

**Email:** sagescience@sagepub.com

If you would like to be added to our electronic mailing list, please send an email to susan_hanscom@sagepub.com.
AN INTRODUCTION TO NATURAL COMPUTATION

Dana H. Ballard

"This is a wonderful book that brings together in one place the modern view of computation as found in nature. It is well written and has something for everyone from the undergraduate to the advanced researcher."

— Terrence J. Sejnowski, Howard Hughes Medical Institute at The Salk Institute for Biological Studies


304 pp. $45

SOFTWARE AGENTS

edited by Jeffrey M. Bradshaw

A comprehensive survey of the state of the art in the design and use of intelligent software agents and in the creation of communication ability between agents.

450 pp. $40 paper

Now in Paperback

ARTIFICIAL MINDS

Stan Franklin

"A highly imaginative and original synthesis of work in the sciences of the artificial and the natural. The book combines the best of cognitive science research with artificial intelligence theorizing, building a bridge between areas usually a chasm apart. I enjoyed it, too!" — Robert Orinstein, psychobiologist and author of The Roots of Self, The Evolution of Consciousness, and Multimind

A Bradford Book • 464 pp., 95 illus. $17.50 paper

SLAVES OF THE MACHINE

The Quickening of Computer Technology

Gregory J. E. Rawlins

Rawlins argues that it is lack of basic knowledge that threatens to make us slaves to computers, and he shows how we can take control of them once more. Amusing, thought-provoking, and packed with information, it will put you "under the hood" of the present-day computer and those of the future.

A Bradford Book • 240 pp. $25

THE EVOLUTION OF C++

Language Design in the Marketplace of Ideas

edited by Jim Waldo

This collection of articles traces the history of C++ from its infancy in the Santa Fe workshop, to its proliferation today as the most popular object-oriented language for microcomputers. Waldo notes in his postscript that in the process of evolving, the language has lost a clearly articulated, generally accepted design center, with no common agreement about what it should or should not be in the future.

279 pp. $27.50 paper

COMPUTABILITY AND COMPLEXITY

From a Programming Perspective

Neil D. Jones

"This is an introduction to the basic concepts of computability, complexity, and the theory of programming languages. The author knows very well all three subjects, has made important contributions to them, has original insights and delightful personal points of view, and overall has good taste. I know of no previous book that provides a comprehensive introduction to all three subjects."

— Christos H. Papadimitriou, University of California at Berkeley

Foundations of Computing series

464 pp., 25 illus. $45

Original in Paperback

GREAT IDEAS IN COMPUTER SCIENCE

A Gentle Introduction second edition

Alan W. Biermann

Presents the "great ideas" of computer science that together comprise the heart of the field. This second edition has new chapters on simulation, operating systems, and networks. In addition, the author has upgraded many of the original chapters based on student and instructor comments, with a view toward greater simplicity and readability.

568 pp., 264 illus. $37.50
Local User Groups

UNIX and LINUX Groups

The USENIX Association will support local user groups by doing a mailing to assist in the formation of a new group and publishing information on local groups in :login, and on its Web site. Full details can be found at: <http://www.usenix.org/membership/LUGS.html>.

At least one member of the group must be a current member of the Association.

Send additions and corrections to: <login@usenix.org>

California
Orange County
UNIX Users Association of Southern California (UUASC)

Colorado
Boulder
Boulder Linux Users Group
Front Range UNIX Users Group

District of Columbia
Washington
Washington Area UNIX Users Group

Florida
Orlando
Central Florida UNIX Users Group
Western
Florida West Coast UNIX Users Group

Georgia
Atlanta
Atlanta UNIX Users Group

Kansas/Missouri
Kansas City
Kansas City UNIX Users Group (KCUUG)

Michigan
Detroit/Ann Arbor
Southeastern Michigan Sun Local Users and Nameless UNIX Users Group

Missouri
St. Louis
St. Louis UNIX Users Group

New England
Northern New England UNIX Users Group (NNEUUG)

New Mexico
Albuquerque
ASIGUNIX

New York
New York City
Unigroup of New York City

Oklahoma
Tulsa
Tulsa UNIX Users Group, $USR

Texas
Austin
Capital Area Central Texas UNIX Users Society (CACTUS)

Dallas/Fort Worth
Dallas/Fort Worth UNIX Users Group

Houston
Houston UNIX Users Group (HOUNIX)
Meets 3rd Tuesday of each month

Washington
Seattle
Seattle UNIX Group

Armenia
Yerevan
The Armenian UNIX Users Group (AMUUG) was founded in December 1996. AMUUG is open to all interested individuals and organizations, regardless of affiliation, without any fee.

Canada
Alberta
Calgary UNIX Users Society (CUUG)

Manitoba
Manitoba UNIX Users Group (MUUG)

Ontario
Toronto Group
Ottawa Carleton UNIX Users Group (OCUUG)
System Administration Groups

SAGE supports local groups. Full listing of group Web sites and other details can be found at:

http://www.usenix.org/sage.locals/

ASUQ
Meets first Wednesday of every third month in Montreal, Quebec.

AZSAGE
Meets monthly in the Phoenix area.

BayLISA
Serves the San Francisco Bay Area.

Back Bay LISA (BBLISA)
Serves the Boston, Massachusetts and New England area.

Beach LISA
A group for system administrators in the San Diego area hosts monthly meetings and the occasional social event.

dc.sage
Serves the Washington D.C. Area.

Dallas-Fort Worth SAGE (dfwsage)
Serves the North Texas area.

$GROUPNAME
Serves the New Jersey area.

EnglishBayLISA
Serves the Vancouver, British Columbia and the British Columbia lower mainland.

Houston Area Sysadmins (HASH)
Serves the greater Houston Area. Join the mailing list by sending a subscribe message to hash-request @tree.egr.uh.edu

Los Angeles Area Group
A group in Los Angeles is being formed. Please contact Josh Geller (joshua@cae.retix.com) if you are interested.

New York Systems Administrators (NYSA)
Serves the New York City area.

North Carolina System Administration Interest Group
Serving central North Carolina, particularly the Triangle Area.

Old Bay SAGE
A group for sysadmins in the greater Baltimore Maryland area.

Seattle SAGE Group (SSG)
Seattle, Washington area.

Twin Cities Systems Administrators (TCSA)
Serving the Twin Cities and surrounding areas of Minnesota.
Fresh Eyes

Have you ever heard that phrase “fresh eyes”? As in: “He looked at his mother with fresh eyes.”

I’ve been thinking about it since my trip to Poland with the USACO team. It took several days for me to be able to see the buildings, people, and general surroundings in Poland without being distracted by the differing details (e.g., different language for signs, different styles of roads, different styles of architecture and generic infrastructure).

After a few days of riding trains and buses, I finally was able to see how the infrastructure was constructed and, at some superficial level, how the country and its economy were run. I wasn’t pleased to find that I had a certain set of prejudices and that my internal weighting functions often put far too much weight on, e.g., the tidiness of a building’s exterior, than on what happens inside the building.

Upon my return home, I find that I am looking at things with a certain kind of “fresh eyes,” having trained them for a different culture for at least a brief time. I ask questions like, “Why in the world did they lay out the sidewalk like that?” I question far more little details than ever before. I even look at my own habits and external appearances (demeanor, etc.) and question them. It’s spooky.

My friends with children of various ages seem to indulge in these same sorts of practices. They gleefully relate amusing tales of their children’s views of the world. They, too, seem to be seeing the world through fresh eyes.

In my home town, Colorado Springs, we have a particularly polarized political scene. I am now viewing that with a certain set of fresh eyes. I am trying to understand why anyone would prefer to be divisive and polarizing rather than trying to unify and search for common ground among a populace. I am pleased as I observe that this kind of polarizing isn’t very strong within our own organization, USENIX.

Have you tried looking at the world with fresh eyes? As I listen to people with differing viewpoints (and I mean really strange, alien, bizarre viewpoints), I try to put myself in their shoes and figure out where they’re headed – and even why. It’s an interesting experiment.

If you’ve never tried, you might give it a whirl sometime.
End the Frustration

A breakthrough in enterprise performance management.

PerformanceWorks™

If trying to manage performance gives you a splitting headache, take the Landmark approach. PerformanceWorks provides the information you need to proactively manage application performance across your enterprise.

Call Landmark today. We’ll prove that it can feel good to use your head.

LANDMARK®
Focused on Performance

www.landmark.com
1-800-488-1111 • 703-802-8000
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 October 7, 1997 and December 9, 1997.