EDITORIAL
Life is Always Full of Surprises

LETTERS TO THE EDITOR
Shell Script Programming from Hank Langeveld
Reply from Jerry Peek

USENIX NEWS
USACO Taps Four for International Olympiad
by Rob Kolstad
USACO: Letters from Students
USENIX PGP Key-signing Service by Greg Rose
Instructions for the USENIX PGP Key-signing Service
by Greg Rose
Board Meeting Summary by Ellie Young
Report on COOTS '96 by Tim H. Harrison
We Want Your Opinion!
New Managing Editor by Ellie Young

SAGE NEWS
Letter from the President by Paul Evans
Call for Nominees for Election to SAGE Board of Directors
Moving Big Data Fast by John Schimmel
Perl Practicum: Network Wiles (Part I) by Hal Pomeranz
From the SAGE Secretary by Timothy Gassaway
SANS '96 Conference Reports
Dear Miss Management by Anonymous
A Different Kind of Networking by Barbara L. Dijker
How Local Groups Can Make a Difference by Carolyn J. Sienkiewicz
SAGE Award Nominations

SAGE FOCUS: BEING A CONSULTANT
Editorial by Tina M. Darmohray
36.8% Overhead or Money, the Bottom Line on Consulting by Steve Simmons
Making the Jump: Moving from Permanent to Contract Employment by Dave Clark

SAGE FOCUS (Cont.)
The Computer Consultant's Workbook—Review by Brent Chapman
Breadth of Vision—A Key to Successful Consulting by Celeste Stokely
Fifty-one Weeks by Mark K. Mellis
My Own Boss by Shawn Instenes

FEATURES
Interview with Andrew Hume by Rob Kolstad
Musings by Rik Farrow
Linkage and Symbolism by Quentin Fennessy
The Cyberminds' Invasion of Earth: Patients and Physicians Online by Emily W. Salus
The Webmaster: Logging in to your Web Pages by Dave Taylor

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

BOOK REVIEWS
The Bookworm by Peter H. Salus
TCP/IP Illustrated, Volume 3—Reviewed by Adam S. Moskowitz
IPng: Internet Protocol Next Generation—Reviewed by George Neville-Neil
Hands-On Netscape: A Tutorial for Windows Users—Reviewed by George W. Leach
UNIX Tamed—Reviewed by Rick Umali
Information Security Policies Made Easy—Reviewed by David L. Oppenheimer

ANNOUNCEMENTS & CALLS
HOT-OS VI
LISA '96
Electronic Commerce

ETC . . .
Local User Groups
Calendar of Events
The closing dates for submissions to the next two issues of *login:* are August 14 and October 9, 1996.
Life is Always Full of Surprises

The big news this week is that I am recovered from my encounter with the medical establishment. It appears that I must confront my fleeting youth (you never hear about 22 year-olds getting kidney stones).

It was on a trip to Taipei, Taiwan, that I discovered that not all systems were “GO.” A quick trip to the doc-in-a-box at the SFO airport (with only 55 minutes to make the Taiwan connection) revealed that I either had a kidney stone (99%) or some super-obscure kidney ailment that no one ever gets (1%). Doctor’s instructions: I’d be a fool if I were to fly to Taiwan. (I queried about not being a fool if I didn’t; he had no comment.)

He gave me a “doctor’s note” to get me to the head of the taxi line as I was rushed to Peninsula Medical Center for high priority X-rays. I should point out that at no time was I in any real pain as so many other kidney stone sufferers are! I was just following standard medical practices to confirm the diagnosis.

The emergency room and hospital personnel were incredibly efficient, and it was great fun to observe all the medical applications of the technology that I studied for so many years—so many neat machines and analyzers.

Pity about the hospital robes. Their efficiency at stripping away all one’s clothes—and dignity—was unparalleled.

Amazingly enough, it took four weeks to be scheduled into the high-tech lithotripter machine (which performs “lithotripsy”). You sit in an elliptical hot-tub with a high-energy sound generator at one focus of the ellipse and your kidney stone at the other focus. Sound waves diverge through the hot-tub away from the sound source then converge back onto the stone, thus shattering it. How convenient that the refractive index of our bodies just about matches that of water. Piece of cake, huh? Not invasive at all!

Well, they perform the procedure under general anesthesia. My recuperation was difficult. Maybe it was because of the other kidney drugs, but I had a heck of a time getting back to this world.

I’m OK now, I guess. It’s amazing how quickly one can go from healthy to not, and then back again.

It’s pretty damn neat that we live in a world with so much gadgetry that we can actually fix things that go wrong. Wouldn’t it be nice if software was so easy to diagnose and repair?

RK
Shell Script Programming

from Henk Langeveld
<Henk.Langeveld@holland.sun.com>

Shell script programming has a reputation for quick solutions, and adequate, but relatively slow performance. This reputation is undeserved, as the traditional shell can perform quite spectacularly when choosing a voluntary confinement: use mostly builtin features. Because the startup overhead of the shell (at least the Bourne and Korn shells)) is quite small, this is a definite win over Perl, for example.

In the Letters to the Editor section of the April 1996 issue, Jerry Peek discusses some techniques to speed up CGI script processing and explains how "[T]he expr(1) program can do all [that] in one step, without the pipe:"

```
browser=`expr "$HTTP_USER_AGENT" :
     \(/{\text{/}}/\)+.*`
```

May I suggest the following piece of powerful sh(1) idiom instead? It only uses builtin shell facilities, thereby removing the need for backquote substitution and its concomitant process forking overhead:

```
_IFS="$IFS" IFS=/
IFS="$_IFS" set -- "$HTTP_USER_AGENT"
browser_name="$1"
```

My point is that traditional shell programming encourages the toolbox approach of UNIX, and there's nothing wrong with that!

Splitting a string on a certain (set of) character(s) is a common requirement when programming in sh(1). I think that applying regular expressions with sed/awk – or even expr – is overkill.

Reply

from Jerry Peek
<jerry@ora.com>

Thanks, Henk. I thought back on why I haven't used IFS, and I remember now. The way it behaves has never made sense to me. For example, with IFS set to a slash (only), here's what happens on SunOS 4.1.3:

```
% /bin/sh
$ _IFS="IFS"
$ IFS=/
$ HTTP_USER_AGENT="Mosaic 1.2 foo/bar"
$ set/\x $HTTP_USER_AGENT
$ echo/$1 x
$ echo/$2 Mosaic 1.2 foo
$ echo/"IFS='IFS',_IFS='$_IFS'"|cat/-t/-v/-e
IFS='/',_IFS='^I$'
```

IFS doesn't include a space anymore, so why does the shell split the "x" into $1? It seems to me that $1 should be "x Mosaic 1.2 foo" and $2 should be "bar". The explanation in the sh(1) manpage was too terse for me, so I've avoided using IFS on random input. I'd love to hear an explanation since using the shell's internal parsing sure is faster than the way I do it.
The 1996 USENIX-sponsored USA Computing Olympiad (USACO) has finished its week-long training camp in Kenosha, Wisconsin at the University of Wisconsin/Parkside on June 15, 1996. Fifteen of America’s brightest pre-college computer programmers tackled programming problems that required solutions that ranged from simple recursive descent all the way through dynamic programming. The participants were chosen from among over 300 students who participated in earlier rounds. They hailed from both public and private high schools across the USA and included:

Daniel Adkins, 11/16, McKinley H S, Baton Rouge, LA
Nick Allen, 11/16, Thomas Jefferson H S for Science and Technology, Alexandria, VA
John Burke, 12/17, North Carolina School of Science and Math, Durham, NC
Russell Cox, 11/17, Delbarton School, Morristown, NJ
Matt Craighead, 10/14, St. Paul Academy and Summit School, St. Paul, MN
James Dang, 12/17, Montgomery Blair H S, Silver Spring, MD
Michael DeSalvo, 11/17, Newman School, New Orleans, LA
Alex Halderman, 9/15, Newton Jr H S, Newtown, PA
Keldon Jones, 12/17, Oklahoma School of Science and Math, Oklahoma City, OK
Shawn Lee, 11/17, Punahou School, Honolulu, HI
Sean Maclsaac, 10/15, Thomas Jefferson H S for Science and Technology, Alexandria, VA
Scott Morris, 12/17, EAST H S, Salt Lake City, UT
Jacob Richman, 10/15, Pennsylvania Home Schoolers, Kittanning, PA
Evan Tsue, 11/16, Stuyvesant HS, New York, NY
Joseph Turian, 10/15, Great Neck H S, Great Neck, NY

The purpose of the USACO is to identify the four USA representatives to the International Computer Olympiad to be held in Hungary in July. Additionally, the staff of four coaches (including Don Piele from UW-P, director; Greg Galperin from MIT, deputy director; Rob Kolstad from BSDI, head coach; and Hal Burch from the University of Missouri-Rolla) provides extensive training and instruction.

This year’s USA representatives to the International contest are: Dan Adkins, Matt Craighead, Keldon Jones, and Joseph Turian. Congratulations to them and best of luck as they go for the gold in Hungary.

Sample USACO Problems

Just to give a sample of the kinds of problems that the USACO students try to solve, consider these two:

**Lab Problem 5:** Place 14 knights on an 8x8 chess board so that every square on the chessboard is attacked by at least one knight. [Hot solution time from this year’s students: 500 milliseconds on a Pentium/100 using Borland C.]
Final Round Problem 4: Dances With Cows [Galperin, et al., 1996]

Editor's note: The USACO uses Wisconsin cows as its theme.

Farmer John is organizing some of his more coordinated cows into a dancing troupe: Mikhail Baryshnicow and his Holstein Hoofers. He has two types of cows in this troupe: Holsteins and Guernseys (the Guernseys are still protesting the troupe's name). Unfortunately, none of them can remember very many dance moves, so he is trying to choreograph in such a manner that they have to remember as few moves as possible.

They know only one type of move: a cow can move from one position in the line to another position (the other cows make room); see the example below. Write a program that will calculate the minimum number of moves to change from one configuration to another; output a minimal length sequence of configurations. If there is more than one optimum set of moves, output only one.

Runtime limit for this problem was 45 seconds on a Pentium/100 using Borland C.

INPUT FILE: dance.in

The input will comprise two lines of letters representing Holstein cows ('H') and Guernsey cows ('G'). The first line will be the beginning configuration; the second line will be the final state that Farmer John wants. There will be no more than 15 cows total.

Example: GHHGGG
          GGGGHG

SCREEN OUTPUT:
A minimal length series of configurations that even Farmer John's forgetful cows can learn to perform.

Example (comments are for exposition):
GHHGGG Initial configuration; number the cows 1 ... 6
GHGGGH Cow 3 moves to position 6
GGGGGH Cow 2 moves to position 4 ... final position

Sample problem data:
HGHGHGGGHG
GHGHGHHHHG
[solution in three moves]

HHHHHHHHGGGGGGG
GHGHGHGHHGGHHH
[solution in seven moves]

HHHGGGHHGHHHHH
GHGHHGHHHGGGGG
[solution in seven moves]

USACO: Letters from Students

[Below are two letters we received from high school students who recently participated in the USACO program which USENIX sponsors. For additional information, see the USACO website: http://usaco.uwp.edu/]

Dear Ms. Young and USENIX,

I would like to express strongly my appreciation of USENIX's appropriation of funds for the USACO program. With your moneys, the staff of USACO (Don Piele, Rob Kolstad, Hal Burch, and Greg Galperin) were able to create not only an intense learning environment, but also, moreover, an enjoyable experience. To temper hours of lectures and coding sessions, discussion groups and programming competitions, the staff were able to bring us participants to movies, to dinners, to ice cream parlors, and even to an amusement park. Logically inserting these recreational activities between periods of deep thought, the staff ensured that all students could learn, without being forced to shuffle off the new information because their heads were spinning with bits and bytes from the previous hours of computing work. Your graciousness and support created an amazing program. I am one of the four students who will represent the United States of America in Hungary. I am indebted to USENIX for giving me the opportunity to compete internationally and to meet students and leaders from around the world. Thank you!

Joseph Turian

Dear Ms. Young,

I just returned from the final round of the 1996 USA Computing Olympiad, where I competed as one of the top fifteen high school computer programmers in the US.

The most meaningful part of the USACO is definitely the final round, a week long training camp held each year at the University of Wisconsin-Parkside. The final round is a great experience because of the people there.

The USACO Director is Don Piele. He created his own international computer programming contest around fifteen years ago, and was the driving force behind the US entry into the International Olympiad in Informatics (IOI). He created the USACO to choose the four best representatives of the United States to attend the IOI. He loves computer programming contests and working with us, and is just a great person to be around.

The USACO Chief of Staff is Rob Kolstad of BSDI. He brings incredible enthusiasm and a vast knowledge base of computer programming problems and tricks. His practical insight into many of the engineering problems of large systems or
large programs is a tremendous help. Outside of lectures and labs, he can be found engaging in many activities with us, ranging from playing ultimate frisbee to discussing life in general.

The USACO Deputy Team Leader this year is Greg Galperin. He and Rob will accompany Don and the four finalists to the IOI this year. Greg, a grad student at MIT, is very experienced in computer programming contests, having enjoyed much success on Harvard’s team to the ACM International Programming Contest. Greg is thoughtful and pensive where Rob rushes in, and many a pleasant lunchtime has been spent listening to them debate various ways to do things with computers. These debates are usually ended by working out the math and figuring, say, that Altavista must keep the entire web index in RAM since hard drive access speeds are just barely too slow.

The final USACO staff member, Hal Burch, is a math, physics, and computer science major at the University of Missouri-Rolla. An incredible programmer, he can optimize just about any code by an order of magnitude or more. One of the highlights of the week was watching him write (and debug) a program from scratch. His skills are really beyond words. One problem whose runtime limit was three minutes he got down to 0.07 seconds. Rob summarized the end of the week critique as “we should clone Hal.”

Then there are the contestants themselves. America’s fifteen best high school age computer programmers are an incredibly diverse group. One is home schooled and can juggle rubber chickens while riding a unicycle (he actually brought these with him to the camp!). Another has spent years on an assembly language program to calculate Euler’s constant to four thousand decimal places in under half a second. A freshman from Pennsylvania enjoys card games and bird watching. Another finalist begins his introductory email message with “Greetings, Earthlings,” and is working on a 2.5 dimensional game and heuristic virus scanners. One contestant from Louisiana has learned all there is to know about algorithms in the past year, and should place extremely high at this year’s IOI. There are more—the group ranges from Hawaii to Virginia, and Minnesota to Louisiana.

The magic of the USACO final round is not whether you win or lose, and not even how you play the game. I have won, lost, played poorly, and played well, and each time I came away feeling the same. The real value of USACO is in meeting this incredible group of people—the fifteen best American high school programmers and an incredible staff. The IOI’s value is exactly the same—you get to meet the 200 or so best high school programmers in the world, and learn that ignoring nationality, everyone is really the same. Fifteen incredibly lucky students experienced the USACO final round this year in Wisconsin; four of them will go on to experience the IOI. And none of this would have been possible without USENIX’s sponsorship. For this, you and all the members of USENIX have our undying appreciation and gratitude. Thank you.

Sincerely,

Russ Cox
1995 USA Team Member 1996 Finalist
rsc@research.att.com
[International Gold Medal Winner—Ed.]
USENIX PGP
Key-signing Service
By Greg Rose
<Greg_Rose@usenix.org>

These two articles follow the Introduction published in the June, 1996, login: Please see the USENIX Web page <http://www.usenix.org> for additional details and late-breaking news.

Statement of Policy

USENIX offers this key-signing service gratis to USENIX members for the purpose of introducing those members to other PGP users. USENIX does not assert any position of authority, and so cannot be considered to be a "Certification Authority". This service shares many of the characteristics of the service provided by a Certification Authority however, and USENIX attempts to adhere to the same guidelines.

Criteria for USENIX's signature

USENIX will sign a PGP public key under the following conditions:

- an individual has been physically present at a USENIX booth at a conference or trade show.
- the individual has presented to a USENIX representative two forms of identification, at least one of which has a picture of that individual on it. Acceptable forms of picture identification include a driver’s license, passport, state or country ID card, or university student ID card, although others may be accepted at the discretion of the representative. The representative has the discretion to refuse any particular item of identification, including those listed. The USENIX representative records the form and identifying serial numbers of those items, and performs a visual check that the identification appears genuine. No follow-up check will normally be performed.
- the individual must sign a statement in the presence of the USENIX representative. Primarily this statement attests that the individual is not misrepresenting his or her identity, and indemnifies USENIX of the consequences of any misuse of the PGP key(s).
- the individual must be a current member of USENIX at the time his or her key is signed.
- the PGP key(s) and user ID(s) which are to be signed must be in the normal form for PGP user IDs, that is:

  User Name <uname@their.domain>

- the “User Name” in the PGP user ID must be a reasonable variation of the name shown on the identification.
- each PGP key to be signed is sent to USENIX via electronic mail.
- the email address in the PGP user ID must be a reasonable variation of the email address from which the key is sent to USENIX.
- A USENIX “shared secret,” consisting of a series identifier, a keyword, and eight secret words will have been given to the individual at the time the ID statement was signed. The email containing the PGP key must include the shared secret in the following form to facilitate automated checking:

  Series: July96
  Keyword: exon
  Secret: glue pant whee koch jacm cosh rose rime

- The electronic mail must be encrypted using USENIX’s current year public key, otherwise the security of the shared secret is deemed compromised and USENIX will not (then or later) sign the key based on this shared secret.
- The electronic mail must also have been signed using the member’s secret key. The fact that USENIX’s shared secret was part of the signed exchange proves that the individual actually has control of the secret key corresponding to the signed public key being signed.

Note that the fact that USENIX receives electronic mail from the address, and sends the reply to it, proves that the key holder can be contacted at that address, but it does not prove that the address belongs to the individual; one could have performed that part of the transaction by forging and snooping.

USENIX’s Key Management

USENIX employees will undergo special training in the concepts of public key cryptography, the use of PGP and other tools used in the key-signing service, and security issues. After this training, they will be asked to agree to the treatment of keys specified in this document.

USENIX has a single master key, which is used only to sign other USENIX keys for operational purposes, and to encrypt escrow copies of the operational keys and revocation certificates for them. Two copies of the above information exist on floppy disks in locked storage, one on site, and one off site. The passphrase protecting the master key and the other information is known to the Executive Director, the President, and one other Director.
This is the PGP information for the master key:

```
pub 1024/2FEA2EF1 1996/04/08 USENIX
Association master key <not-for-mail>
Key fingerprint = DB A7 50 99 66 E4 8A A9 80
 B2 D9 E2 FE DA 00 5E
```

PGP does not currently support expiry dates for keys. USENIX will, instead, create "operational" keys which are valid (according to this statement) for particular calendar years. The user ID for these keys is of the form "USENIX yyyy".

This is the PGP information for the 1996 operational key:

```
pub 1024/969DF939 1996/04/08 USENIX 1996
<pgp@usenix.org>
Key fingerprint = E5 AE EC CE 8F 69 63 DF 68
 C5 BF 4B 3C F3 FC 11
```

Signatures on members' keys will be made using annual signing keys, which have a World Wide Web reference to a statement about what the signature implies, instead of an electronic mail address.

This is the PGP information for the 1996 signing key:

```
pub 1024/F7A1F72B 1996/04/08 USENIX 1996 Signature
<http://www.usenix.org/pgpsig.html>
Key fingerprint = 82 EE 99 E2 FB EF 5D 78 8D D8 19 42
 A2 E1 DF 1E
```

Keys for the next calendar year will be created, signed, and propagated via keyservers, in time for this fingerprint information to be published in the last ;login: newsletter of the preceding year.

The key for a previous calendar year will be honored for incoming correspondence until the end of January of the following year. Following that time, the online copies of the secret key will be discarded, and further correspondence using that key will be returned with a request to use the current key. Operational keys will not be revoked unless the key is considered compromised.

In the event that an operational key is thought to be compromised, it will be revoked. Revocation certificates will be sent to public keyservers, and mail using the key will be returned along with a copy of the revocation certificate. The revocation will be announced in ;login: along with the details of a replacement key.

Operational keys will only be used in the USENIX offices, and the keys themselves will only leave the offices on off-site backups.

When there is a need to use PGP outside the office, temporary keys will be created. Employees and associates of USENIX will be encouraged to have their own keys, and will be treated as members of USENIX for the purposes of key signing.

Treatment of keys signed by USENIX

USENIX will sign a key initially according to the criteria described above. The signature will be from the current operational key. A copy of the signed key will be returned to the member by electronic mail to the address in the primary user ID on the key. The key will not be sent to any other address, nor to a key server. The member may of course forward the signed key anywhere they wish.

USENIX will record all keys with current signatures. A World Wide Web URL (http://www.usenix.org/cgi-bin/pgp-key.cgi) implements an interface allowing a query with respect to specific keys. This interface will return one of:

1. The member's key, with an attached signature from the current signing key, or
2. A clearsigned document stating that USENIX's signature on the key has been revoked, the reason for this revocation, and (when supplied by the member) a revocation certificate for the key itself, or
3. Nothing at all.

The behavior will change from 1 to 2 within two working days of notification of a compromise of the key. The behavior, once at 2, will stay that way until further notice.

When a new operational key is introduced, around the beginning of the new year, a check will be made that the member is still a current member of USENIX. If they are, and no notification of revocation has been given, a random shared secret will be generated and sent (encrypted and signed) to the member's email address, with a request that it be returned, encrypted to USENIX and signed by the member. A correct response proves that the individual can still be contacted at that electronic mail address (legitimately or otherwise), and still has control of the key. The key will be re-signed with the current signature key and sent to the member as above. If membership has lapsed, the key will be dropped from the database and the behavior of the Web page will change from 1 to 3.

Note: USENIX's Web page interface is a part of the key-signing service, and not a general purpose PGP Key server. Only members' keys which have been signed using this service will be available using it.
Instructions for the
USENIX PGP
Key-signing Service

by Greg Rose
<Greg_Rose@usenix.org>

What to do to get your key signed

The piece of paper issued by USENIX as a shared secret contains a secret set of words, which is the only connection between you and USENIX for the purposes of getting your key signed. Until your key is signed, this piece of paper is as valuable to you as USENIX’s signature on your PGP key eventually will be, so keep it secret! After USENIX has signed your key, it is worthless, and you can do what you want with it.

These instructions are step-by-step and assume no prior use of PGP at all. Depending on your level of current use of PGP you will be able to skip some of the initial steps, but you should know which ones. We also assume you have at least electronic mail access to the Internet. Things are somewhat different using Mac-PGP; Valerie Polichar has provided me with translations for the Macintosh, which I greatly appreciate.

We use the following example in the rest of this document:

You are:
Fred Nurk <fred@nurkkorp.com>

The shared secret we give you looks like:

Series: July96
Keyword: exon
  glue pant whee koch jaco cosh rose rime

When you receive it, it should be folded so that you can’t read the 8 words, and sealed in a way that would make tampering fairly obvious.

1. Obtain PGP

Anyone can get and run PGP. But who you are makes a difference in how to get it. The FAQ for the subject of “How to get PGP” is available on the WWW as http://www.csn.net/~mpj, or by FTP as ftp://ftp.csn.net/mpj/getpgp.asc. The short answer, though, is:

Commercial use in the US:
mailto:viacrypt@acm.org
Non-commercial use in the US:
ftp://net-dist.mit.edu/pub/PGP/

Outside the US:
(or many others, see FAQ).

2. Install PGP

Follow the instructions given with the version of PGP you obtained. These vary wildly depending on which one you got (i.e. your hardware) but all are good and easy to follow. Please read the documentation that comes with it. There is a FAQ for PGP (from the newsgroup alt.security.pgp), which might help with any problems you have, at ftp://ftp.prairienet.org/pub/providers/pgp/pgpfaq.txt or from rtfm.mit.edu in the normal manner.

3. Make Your PGP Key Pair

The command for this is:

pgp -ks fred@nurkkorp.com

and you will be prompted for everything else you need. As there is no real advantage to using short keys, we recommend 1024 bits for your first key. Any larger may not interoperate with people still using older versions of PGP. You should keep to the recommended format for your PGP user ID, as in the example above.

On the Macintosh, double-click on the MacPGP icon to start PGP. Click once in the signature window to start the program. Select “Generate Key” from the Key menu; follow menus.

4. Sign Your Own Key

This is important. Until you have done this, a nasty hacker who managed to steal your public key file could edit the user ID you provided. Once you have signed it yourself, it is effectively read-only, as any alteration will be detected. You do this with the command:

pgp -ks fred@nurkkorp.com

Newer versions of PGP (2.6.3 or greater) do this automatically, in which case you can skip this step.

On the Macintosh, select “Certify Key...” from the Key menu. Open your pubring.pgp. Double-click on your own key to certify. In the next window, double-click on your own key to select the certifier.

5. Get USENIX’s Public Key

There are a number of ways to get USENIX’s public key. The best is to get it from our WWW homepage, http://www.usenix.org/pgpkey.asc. It is also available from a keyserver. To do this send electronic mail:
make it (actually, they) will be returned by mail. Extract the bit from the reply which starts with something like:

```
----- BEGIN PGP PUBLIC KEY -----
```

into a file (say `usenix.asc`). Tell PGP to add it to your public key file with the command:

```
pgp -ka usenix.asc
```

On the Mac, save the key to a file (this may happen automatically if you use Eudora). Then select “Add Key...” from the “Key” menu.

Then you want to check that you really did get USENIX’s key, and not some dummy key put out there by a malicious rival. This command will display the fingerprint of the key, which should match exactly with this:

```
pgp -kvc pgp@usenix.org
```

```
pub 1024/969DF939 1996/04/08 USENIX 1996
Key fingerprint = E5 AE EC CE 8F 69 63 DF 68
 C5 BF 48 3C F3 FC 11
```

On the Mac, select “Fingerprint key...” from the Key menu. Double-click on the USENIX key you just added. The fingerprint will be displayed in the “PGP Messages” window in the background.

The “key fingerprint” is a checksum of the information in USENIX’s key, and cannot be forged. You will find it printed in our newsletter, and since we’ve handed you this piece of paper with the secret, you know you have the right key if the fingerprint matches.

Since you know that USENIX’s key really does belong to USENIX, you can sign it to attest to that fact. The command for that is:

```
pgp -ks pgp@usenix.org
```

On the Macintosh, select “Certify Key...” from the Key menu. Open your `pubring.pgp`. Click on your key (a checkmark should appear on the far left). Click to check the box by “Asciiify the output”. Now click on “OK”. Enter a file name for the key which ends in `.asc` (e.g. `mykey.asc`).

This will create a file called “`mykey.asc`”, with some unintelligible stuff in which represents your key.

### 7. Prepare Your Mail to Us

Edit the file “`mykey.asc`” which you just created. Insert the following information before the first line, but be careful not to modify anything in the part PGP put there (even a single character change will invalidate your public key).

```plaintext
Name: Fred Nurk
E-Mail: fred@nurkkorp.com
Series: July96
Keyword: exon
Secret: glue pant whee koch jacm cosh rose rime
```

This is almost everything we need to know to be able to sign your key. It is still sensitive information, though. You must be careful how you send it.

### 8. Encrypt and Sign it

PGP allows you to digitally sign this information, and also to protect it during its transmission to USENIX. That is, after all, the point of PGP. Use the command:

```
pgp -east mykey.asc pgp@usenix.org -o info.asc
```

The flag characters mean “Encrypt, Ascii-armor, Sign, Text-input”. You will be left with a file called “`info.asc`” which contains a (bigger) blob of unintelligible stuff.

### 9. Email it to USENIX

However you do it with your mailer is up to you, but at this point you should send the file “`info.asc`” to the address:

```
pgp@usenix.org
```

You don’t need to send it in any special way, since PGP has already done the equivalent of uuencoding it. Some amount of preprocessing will be done automatically. We will probably handle it if you send it to `office@usenix.org`, but try to save us both some trouble, OK?

### 10. Get Your Signed Key Back

Sometime later, you will get your PGP key back, and it will be signed by USENIX according to our policy. Allow some
time for this to happen; the office staff will batch up the work so they are not continually being interrupted. Save the contents of the return mail message to a file, let's call it "t.asc". Add our new signature to your keyring with the command:

```
pgp -ka t.asc
```

## 11. Other Things You Might Want To Do

You might want to extract your key (as you did in step 6) and send it to a keyserver. This is simple, just send mail to

```
pgp-public-keys@keys.pgp.net
```

with a subject of "ADD", and your extracted key as the body of the mail.

---

**USENIX Board Meeting Summary**

*by Ellie Young*  
*<ellie@usenix.org>*

Below is a summary of the actions taken at the meeting of the USENIX Board of Directors held on May 17-18, 1996 in Fairfax, VA.


### Futures

There was a lengthy brainstorming session to identify breaking areas in technology so that new workshops, invited talks, tutorials, and services could be planned. The group also spent time discussing constituencies and affinity groups, and how USENIX/SAGE would best serve them in the future.

### Student Research Projects

It was decided to allocate $50,000 for student research projects this year. A proposal from Professor Victor Yodaiken of New Mexico Tech., to fund $14,000 for student stipends to do operating systems research was approved. (The students will be working on the Linux PowerPC port, adding realtime capabilities to Linux, and making general improvements on kernel code design in speed and reliability.) It was also decided that future proposals will be sought and would be evaluated by the scholastic committee.

### EurOpen

Lori Grob will attend the EurOpen governing board meeting in June. She will convey USENIX's willingness to help the technical community in Europe and solicit proposals for how we might do so.

### Lifetime Honorary Member

In acknowledgment of her many years of service to USENIX, Debbie Scherrer was made an honorary lifetime member of USENIX.

### Conference Registration Fees

In keeping with past policy to raise fees in small increments in order to keep up with rising costs, it was agreed to raise the technical conference fees by $10 and tutorial fees by $15 effective January 1997.

### Committees/Liaisons

Eric Allman was appointed as the USENIX board representative to the SAGE Board. Peter Honeyman will serve as liaison to the USELINUX conference organizers. The following committees were formed and/or new members were appointed to fill the posts held by previous board members:

- Executive: Hume, Allman, Grob, Johnson
- ;login: editorial: Allman, Darmohray, Zwicky, Kolstad, Farrow
- Prizes and Awards: O'Dell, Rose, Seltzer, Groundwater, Ritchie
- Scholastic Services: Honeyman, Seltzer, Grob, plus one other to be announced
- STG Committee: Hume, Allman, Geer, Zwicky
- Tutorial Review: Klein, Honeyman, Geer, Seltzer, Farrow, Welch

### Next Meeting

It will be held in Chicago (at the site of the LISA Conference) on September 28-29, 1996.
The technical program at COOTS ’96 covered a wide range of topics relating to advanced OO research and development. Session topics at the conference covered C++, CORBA and Distributed Objects, Tools, Patterns, Object-Oriented Frameworks and Components, and Distributed Languages. Twenty presenters came from all over the world including Canada, England, Israel, Germany, Switzerland, and the United States. Attendance at the technical sessions was high, over 230, which may have been due to the quality of the technical sessions or to the fact that Toronto had rainy weather during much of the conference.

This year’s keynote address was by Dr. Dave Thomas, who is the president of Object Technology International (OTI) in Ottawa, Ontario. Dave’s talk was titled “Research and Industrial Experience on the Road to Object and Component Utopia.” In his talk, he alternatively praised and preyed upon nearly every OO buzzword or fad that has surfaced in the past decade. Dave began by listing all the technologies that OTI has leveraged while working towards the completion of their “Object Utopia.” Object Utopia is a ten year project aimed at addressing real problems faced in distributed object computing, many of which are still not solved by contemporary hardware and operating systems. To address these issues, Object Utopia has adopted a virtual machine approach with a shared memory architecture for IPC.

After describing the many positive contributions of the OO community, Dave proceeded to debunk much of the buzzword-laden technology hype that tempts today’s software developers. He pointed out that: “Just-In-Time compilation doesn’t solve everything. Type safety doesn’t improve code reliability or correctness. Metrics and Code Managers are for programmers, not management.” The only event at COOTS more controversial than Dave’s talk was having lunch with Don Box (woe to the Win32 and DCOM non-believers ;-)).

Dave concluded his keynote address with a recommendation for educational institutions. After suffering through automata courses recently, it was refreshing to hear him say that students should build real applications and should not be subjected to software “methodology” or abstruse theory. Dave is a strong proponent of the “Internet University,” which is good news for students, but may be a threat to those faculty who prefer to remain safely ensconced in their Ivory Towers.

**C++ Session**

Chaired by ANSI/ISO vice chair Josee Lajoie, IBM Toronto.

COOTS evolved from the seven-year legacy of the USENIX C++ conference. Therefore, it was fitting to kick off the initial technical session with a set of papers on C++. The first presentation was by Sara Porat on “Compiler Optimization of C++ Virtual Function Calls” (Porat, Bernstein, Fedorov, Rodrigue, and Yahav). Sara and her colleagues are working hard to squeeze out the few inefficiencies left in C++. Their approach involves static global analysis of C++ application code that can result in one of two optimizations. “Unique Name optimizations” replace virtual function calls that have only one existing destination with direct function calls. The “Single Type Prediction” approach replaces virtual function calls with conditional expressions that optimize for the most frequent function destination. Tests of various benchmarks show up to 43% increases in performance.

The next talk was by Keith Loepere from the Open Software Foundation on “Composing Special Memory Allocators in C++.” Memory management is arguably the most feared and loathed aspect of C++ (providing endless strawman fodder for the Java community). Keith gave several examples that motivate the complexity of C++ memory management, including special memory allocators for variable sized objects, storage recycling, real-time memory constraints, and allocation from special memory areas (e.g., shared memory). Unfortunately, because of the freedom that C++ compilers have to lay out the storage of derived objects, operator new and constructors cannot pass data through object data members. The approach described by Keith uses a special allocation record for each object to store the data needed to properly initialize the object. The techniques described used to provide the MK++ microkernel with variable sized buffer objects, buffer recycling for network memory management, and memory resource accounting.

Mark Addesso from the North American division of Software AG gave a talk on “Building Independent Black Box Components in C++.” Mark describes a design strategy for building truly reusable components. These “object-level components” have only one parent, and have no interaction with other components other than well defined input and output “ports.” This data flow style of design attempts to remove the intercomponent couplings that tend to result in reuse-inhibiting object webs. The design techniques described were used to develop “Esperant,” Software AG’s query and reporting tool. The experience showed that, although the design model could not be used pervasively, the resulting black box components simplified development and mainte-
nance by adding structure to the design and supporting unit testing of components.

**CORBA and Distributed Objects**

Chaired by Steve Vinoski, HP in Chelmsford, MA, and C++ Report columnist.

Jennifer Hamilton from IBM Toronto Laboratory started off the “CORBA and Distributed Objects” session. Her paper, titled “Interlanguage Object Sharing with SOM,” won the COOTS ’96 Best Student Paper award. Jennifer’s talk focused on the release-to-release binary compatibility (RRBC) and the interlanguage object sharing qualities of IBM’s System Object Model (SOM). Because of the static nature of C++, changes to library objects frequently require recompilation of client applications. SOM fosters RRBC by adding proxies to object implementations so implementations can vary independently from the interfaces. SOM object interfaces can be defined using OMG IDL or DirectToSOM (DTS) compilers. DTS compilers allow interfaces to be defined in the native language (DTS currently supports C++ and OO COBOL). SOM also provides a unified object representation so that object implementations can be used by clients written in C, C++, Smalltalk, and OO COBOL. With the combination of the language-independent SOM object model and DTS compilers, it is possible to access C++ class libraries from multiple languages.

Jose Bernabei from Sun Microsystems Laboratories presented “Extending a Traditional OS Using Object-Oriented Techniques” (Bernabei, Matena, and Khalidi). Jose described the Solaris MC operating system, which provides applications with a “single-system image” of a cluster of servers. To implement services such as the distributed memory and file systems, Solaris MC uses a communications infrastructure based on the CORBA architecture with the addition of XDoors and Handlers. The Solaris MC ORB uses an extension of the Solaris “Doors” IPC mechanism to provide fault-tolerant communications. Solaris Doors were derived from the Spring OS, which was described in the previous COOTS conference. Handlers are responsible for maintaining reference counts for distributed objects so that unused resources can be freed. The developers of Solaris MC have also specialized a bulk I/O handler that reduces the amount of data marshalling overhead in order to increase the throughput of the distributed file system.

Ram Kordale from the Georgia Institute of Technology talked about “Object Caching in a CORBA Compliant System” (Kordale, Ahamad, and Devarakonda). To help alleviate the performance costs of distributed computing, Ram and his colleagues are exploring the benefits of caching distributed CORBA objects. Ram described Flex, a distributed object caching system that addresses the cache coherency issues inherent to the domain. One unique quality of Flex is its support for multiple levels of cache consistency. Applications can use strong consistency or just causal consistency where newly cached objects are guaranteed to reflect all “causally preceding events.” Performance tests run on Flex show significant gains for applications that exhibit certain locality of reference properties.

The paper titled “Asynchronous Notifications Among Distributed Objects” (Aahlad, Martin, Marathe, and Le) was presented by Bruce Martin of SunSoft, Inc.. Bruce described the SunSoft implementation of Event Channels, which is an OMG CORBA Services standard for providing asynchronous and anonymous communication between suppliers and consumers. To resolve the shortcomings of the canonical CORBA synchronous request/response model of distributed computing, Event Channels allow peers to “push” typed or untyped events asynchronously to one or many receivers. Similarly, Event Channels allow peers to synchronously pull events from one or more senders. Bruce discussed some of the implementation issues of SunSoft’s Event Channel, as well as performance results that measured the throughput and latency of their Event Channel for both transient and persistent events.

**Tools Session**

Chaired by Doug Lea, SUNY, Oswego

The Tools session started with Sreenivasa Viswanadha from the State University of New York. He presented “Preliminary Design of ADL/C++ -- A Specification language for C++” (Viswanadha and Sankar). Sreenivasa described their research on the the ADL set of tools for specifying and testing C and C++ programs. Like CORBA IDL, ADL specifications are similar to C/C++ syntax (including inheritance, virtual functions, and exceptions). Therefore, users can describe the input-output behavior of functions without having to learn an unfamiliar specification language. ADL is used to define postconditions for functions and can take test input and results from a function to test for correctness. In addition to specifications for functions, the group is exploring the possibility of behavior specifications for C++ classes. With the completion of ADL/C++ and plans for ADL/Java and ADL/IDL, the group is researching powerful tools to improve the correctness of software.

Pornsiri Muenchaisiri presented “Software Composition with Extended Entity-Relationship Diagrams” (Muenchaisiri and Minoura). Pornsiri’s research group has developed a tool for graphically designing and building software systems. An extension of the entity-relationship approach to software design, Pornishiri described the Entity-Relationship Software Development Environment (ERSDE). In the ERSD, programmers can use object or “entity” subclassing, compo-
sition, and relationship metatypes to build domain-specific components. Applications can, in turn, be constructed from the reusable domain-specific components. ERSDE has been used to graphically constructing network and factory simulators. Currently the group is enhancing the development environment to allow applications to be composed into larger applications.

The paper titled “A Measure of Testing Effort” (McGregor and Srinivas) was presented by John McGregor from Clemson University. John described a code metric designed to estimate the effort involved in testing object-oriented software. This metric can be used to create a new measure termed the “visibility component” (VC) of OO methods, along with a composite VC for classes composed of methods. This metric is designed to be applied early in the design phase in order to understand the costs of developing the software. More specifically, the goal is to provide an estimation regarding the likelihood that an object will, when tested, reveal its faults. The authors conducted several experiments to verify their hypothesis, the results of which are provided in the article. They hope that this metric, coupled with others, will provide a tool for managers to gain insight into the progress of their respective projects.

**Patterns**

Chaired by Doug Schmidt, Washington University.

Doug Schmidt chaired the Patterns session and startled the COOTS attendees by regretfully informing us that Robert Martin could not attend. Apparently, he’d been accosted at the Canadian border by “OLE-sniffing dogs.” Unsure of the veracity of the story, the audience was encouraged to check the proceedings for Roberts paper on “Design Patterns for Dealing with Dual Inheritance Hierarchies in C++.”

Using up the bulk of Robert’s allotted time, Silvano Maffeis, from Olsen & Associates in Zurich, presented “The Object Group Design Pattern.” Silvano explained the Object Group pattern as an abstraction over reliable group communications. Object Groups are useful for a variety of software areas such as fault-tolerance, groupware, and mobile computing. Clients of Object Groups use a single reference to initiate requests to the entire group. The communication subsystem is responsible for providing “Virtual Synchrony” in delivering requests to all group members. Following requests, clients are able to wait for responses from all or some of the group. Silvano described how the Object Group pattern could be implemented as a CORBA Basic Object Adapter (BOA). He also mentioned his own implementation of the Object Group pattern that is a BOA used in the Electra ORB, which encapsulates procedural reliable distributed toolkits such as ISIS.

Harald Müller, from Siemens AG in Austria, described “Pattern Languages for Handling C++ Resources in an Exception-Safe Way.” Harald discussed several patterns that help to put the “handling” back in “exception handling.” The “Ensure Shut-down-ability” pattern describes rules for building objects that can always free resources, even in the face of exceptions. The “Hide damaged resources till destruction” pattern explains how to develop objects that can still be used after parts of the object have become damaged. Harald also presented techniques that can prevent resources from becoming damaged due to exceptions. These patterns require assigning to and passing between objects the responsibility of a resource (e.g., freeing the resource). Using management objects such as `C++ auto_ptr<T>`'s with patterns such as “Responsibility transfer is swap,” objects can define operations that correctly manage resources when exceptions are thrown.

**OO Frameworks and Components**

Chaired by Jim Waldo, JavaSoft Inc.

Starting off the Object-Oriented Frameworks and Components session was Kai-Uwe Mätzel from Union Bank of Switzerland. Kai-Uwe presented “The Any Framework, A Pragmatic Approach to Flexibility” (Mätzel and Bischofberger). The Any Framework provides a language-independent, garbage collected, data representation mechanism. Any objects are dynamically extensible to allow run-time evolution of object contents and can be streamed into human-readable strings for distribution and debugging. Any objects are grouped into AnySoups that can be queried by the Object Database Standard object query language (OQL) for object retrieval. The Any Framework was used extensively in the development of the Beyond-Sniff system for distributed, multi-user software development. Most notably, the Python interpreter augmented Beyond-Sniff services with a scripting language by using the Any Framework to directly manipulate the data objects used by the C++ services.

Douglas Schmidt from Washington University discussed the “Design and Performance of an Object-Oriented Framework for High-Speed Electronic Medical Imaging” (Pyarali, Harrison, and Schmidt). Doug described the Blob Streaming framework for efficiently transferring Binary Large OBjects (BLOBs) in a distributed medical imaging system. The framework allows medical imaging applications to operate on BLOBs (e.g., images) independent of BLOB location and type. Using the uniform BLOB Proxy interface, applications can manipulate images from memory, a file, or a database. The communications infrastructure of the framework allows applications to transparently access BLOBs from remote hosts, as well. BLOB Streaming integrates sockets and CORBA seamlessly in order to provide performance and flexibility through the framework. The framework has been thoroughly tested and profiled—its throughput performance is
close to that of using sockets directly. BLOB Streaming is currently being used in Project Spectrum at the Washington University School of Medicine and BJC Health System.

James Miller from the University of Kansas presented "Class Relationships and User Extensibility in Solid Geometric Modeling." James spoke on the cryph Solid Modeler and the use of Object-Oriented techniques to maximize system flexibility and extensibility. Two key requirements in Computer-Aided Design and Manufacturing applications, such as Solid Modeling systems, are reuse and specialization of geometric software, and user extensibility of modeling primitives. The cryph modeler was developed in C++ to reuse legacy C utilities and to allow class relationships between the line, plane, conic, and quadric geometric forms. Users can also extend the modeling primitives via Macro Solids for frequently used geometric features. Though the cryph framework has been decoupled from platform-specific image renderers, the developers have constructed independent renderers for displaying the cryph geometric models.

**Distribution Languages**

Chaired by Daniel Edelson, IA Corporation

The last technical session category was Distribution Languages. Jim Waldo from JavaSoft presented "A Distributed Object Model for the Java System" (Wollrath, Riggs, and Waldo). Jim explained the Java Remote Method Invocation (RMI) system that adds distribution to the Java object model. Like CORBA, Java RMI supports remote object invocations, but unlike CORBA, Java RMI is specific to the Java programming language. This allows Java RMI to take advantage of Java features such as garbage collection, downloadable code, and security managers. Since there is no separate IDL interface, Java RMI clients can use any Java types while using proxies to invoke remote operations on server implementations. The Java RMI architecture provides a client stub and server skeleton layer to perform parameter marshalling and operation demultiplexing, a Remote Reference Layer to implement various invocation polices (e.g., unicast and multicast), and a Transport layer to abstract from the communication mechanisms. Current work on Java RMI includes support for server activation and replication. JDK 1.1 will contain the first Java RMI release.

Following up on Jim Waldo's talk, Roger Riggs from JavaSoft explained the “Pickling State in the Java System” (Riggs, Waldo, Wollrath, and Bharat). Pickling is the process of serializing Java objects for storage in databases or transferring across process boundaries. When a Java object is written to object output stream, the stream first serializes the object, and then traverses the objects references to other objects. The whole graph of objects is pickled to the stream and can be conversely unpickled preserving the inter-object references. Each object is pickled with a unique Fingerprint to ensure that the serialized object matches the class available during unpickling. A Java class can also specify how it is to be pickled and unpickled so that special purpose data (e.g., local operating environment) can be included in the object's serialization. Pickling is specifically useful in the context of Java RMI where complex data structures involving multiple objects can be transferred with a single method parameter.

Gregory Wilson from the IBM Center for Advanced Studies in Canada presented “Smart Messages: An Object-Oriented Communication Mechanism for Parallel Systems” (Arjomandi, O'Farrel, and Wilson). Gregory described the Smart Message mechanism for remote method invocation. To perform remote object operations, clients store method pointers and parameters in smart messages that are sent to the server. When the server receives the smart message, it simply invokes the message that "knows" the object, method, and parameters to use. Smart messages leverage C++ features such as polymorphism and templates to simplify use and ensure typesafe parameter marshalling. Smart messages are used internally by the ABC++ class library for creating and invoking operations on remote objects. ABC++ is C++ library developed by the authors for distributed and concurrent object computing. ABC++ combines smart messages and futures in order to allow asynchronous communications between active objects in a type-safe and portable manner.

The last technical presentation was "Highly Concurrent Distributed Knowledge Objects" (Clark and Wang) presented by Keith Clark from Imperial College, London. Keith described DK_Parlog++, a combination of Concurrent Logic Programming and OOP. DK_Parlog++ is an Object-Oriented extension to Parlog where classes and their methods are implemented as Parlog processes. Clients can invoke remote operations synchronously or asynchronously, and server objects can service multiple method invocations simultaneously providing a high degree concurrency. DK_Parlog++ also supports static and dynamic knowledge methods that are specified in Prolog and can return multiple results from each invocation. Derived classes can inherit, override, or differentially inherit the superclass knowledge methods. The DK_Parlog++ language is fully implemented and is currently being used at Imperial College.

**Concluding Remarks**

This year's technical sessions at COOTS were a rich sample of advanced research and development work in object-oriented technologies and software systems from around the world. The talks and tutorials at the conference underscored how Object-Oriented techniques have become an integral part of solutions to many of today's real problems.
strongly encourage everyone to pick up a copy of the conference proceedings to check out each of the papers in more detail. Hardcopy proceedings can be obtained from USENIX for a nominal fee. For USENIX members, the papers will be available online at

Incidentally, due to the success of this year’s COOTS, USENIX has begun planning next year’s event. The program chair for COOTS ’97 will be Steve Vinoski <vinoski@apollo.hp.com> and the tutorial chair will be Doug Schmidt <schmidt@cs.wustl.edu>. Although the ink isn’t dry on the contract yet, it appears that COOTS ’97 will be held in Portland, Oregon June 16-19, 1997.

We want your opinion!

USENIX is considering producing a CD-ROM with all 28 conference proceedings from 1993-1996. However, we do not want to undertake such a project if it does not meet your needs.

Please tell us if you think this is a good idea or not: Fill out the survey on our web site: <http://www.usenix.org>.

There will be a pointer in the “what’s new” section on the home page.

The survey will be available until September 10.

New Managing Editor
by Ellie Young
<ellie@usenix.org>

After 6 years of handling publications tasks for this newsletter and the USENIX proceedings, Carolyn Carr has decided to move on and pursue new ventures in education and the arts. I am pleased to announce that Pennfield Jensen has been hired to manage the publications and online services program (and anything else we can think of!) Penn has over 20 years experience in printing and publishing. He has worked extensively with publishers in four-color graphics and large-document production, especially books. A cofounder of the San Francisco Bay Area Book Festival, an annual event that attracts thousands each year for a weekend of exhibits, readings and workshops, he has also chaired or co-chaired numerous panels on desktop publishing, multimedia and online publishing. He has produced books for many publishers, including Peachpit Press, Sybex Computer Publishing, KQED Books, Chronicle Books and many others. His most recent projects have included The One Bowl Cookbook (David Barich/Chronicle Books) and From the Recipe Files of the C.I.A., The Culinary Institute of America, using digital color photography. Long a proponent and practitioner of computer-based publishing, his skills and experience are a welcome addition to the crew here at USENIX.
SAGE, the System Administrators Guild, is dedicated to the advancement and recognition of system administration as a profession. SAGE brings together system and network administrators for:
• professional and technical development,
• sharing of problems and solutions,
• communicating with users, management, and vendors on system administration topics.

SAGE NEWS EDITOR
• Tina Darmohray
  <tmd@usenix.org>

SAGE BOARD OF DIRECTORS
• Paul Evans, President
  <ple@usenix.org>
• Tim Gassaway, Secretary
  <gassaway@usenix.org>
• Barb Dijker, Treasurer
  <barb@usenix.org>
• Helen Harrison
  <helen@usenix.org>
• Bryan McDonald
  <bymac@usenix.org>
• Hal Miller
  <holm@usenix.org>
• Kim Trudel
  <kim@usenix.org>

SAGE WORKING GROUPS
GROUP
sage-certify
sage-edu
sage-ethics
sage-jobs
sage-local
sage-online
sage-policies

CHAIR
Paul Moriarty
Ron Hall
Hal Miller
Tina Darmohray
Rene Gobeyn
Pat Wilson
Lee Damon

YOU CAN CONTACT THESE GROUPS VIA EMAIL AT
<group-name@usenix.org> for example,
<sage-certify@usenix.org>.

SAGE DISCUSSION GROUPS
sage-security
sage-managers
sage-outreach
sage-pt
sage-solo

SAGE BOFS
sage.board
sage.bof.women

SAGE ONLINE SERVICES
Email server:
majordomo@usenix.org
FTP server:
ftp.sage.usenix.org
WWW URL:
http://www.sage.usenix.org

SAGE SUPPORTING MEMBERS
Bluestone, Inc.
Enterprise Systems Management Corp.
Great Circle Associates
Pencom Systems Inc.
Southwestern Bell
Taos Mountain

SAGE NEWS

Letter from the President
by Paul Evans
<p@synopsys.com>

I have been focusing deeply on the problems of managing system administration since becoming a manager in my group in January. Having spent ten years as a front-line system administrator, I now find myself spending a lot of time trying to explain what things look like from the “other side” to system administrators. Some of them want to become managers themselves, others fervently hope to avoid doing so, but both groups share two common sets of concerns: They are frustrated by their interactions with managers, and they spend a lot of time thinking about how to influence the decisions of their managers, and those of other managers in their organizations.

I believe that in the long term the only way for system administrators to be happier with their managers is for more system administrators to become managers. But until that happens, system administrators are going to have to learn how to communicate with managers who share neither their technical knowledge nor their professional values. Bill Howell has observed that unless you’re the CEO of a company, your job is to make your boss’s job easier. One of the most important ways you can make your boss’s job easier is to supply compelling information about the issues facing managers—compelling not from the point of view of other system administrators, but from the point of view of those making decisions.

One of the goals of SAGE is to help system administrators present their managers with this kind of compelling information, and thus to help system administrators make their bosses’ jobs easier. The SAGE Jobs Booklet was an excellent example of information that makes it easier to explain system administration to managers who are not system administrators. We are pleased to be bringing out later this year another publication, writing policies for computer sites. Other publications, including one on “Security: What Your Management Needs to Know,” are in early stages of preparation.

Most system administrators experience a pretty direct connection between the quality of their professional lives and the quality of their managers’ decisions. One of the ways SAGE hopes to improve the professional quality of life for its members is to give them the tools to positively influence the quality of these decisions. We’re hoping that the upcoming additions to the publication series will be valuable additions to the system administrator’s toolkit.

Call for Nominees for Election to SAGE Board of Directors
by Kim Trudel
<k@usenix.org>

SAGE is accepting nominations for three new members of its Board of Directors. The nominating period will close October 8, at noon, PST. Anyone interested in running for the SAGE board should send his or her name and telephone number and a brief statement to the nominating committee via email at:
sage-nomcom@usenix.org.
You can also send US Mail to:

SAGE Nominating Committee, care of
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

In this election, directors will be chosen for two-year terms beginning January 1, 1997. They will join returning Board members Paul Evans, Barb Dijker, Helen Harrison, and Tim Gassaway. The SAGE Board chooses its own officers after each annual election. The Board meets in person twice per year, usually at LISA and at one other USENIX conference, and has other meetings via teleconference.

There will be a candidates forum at the LISA '96 conference, to be held September 29-October 4, 1996, in Chicago, Illinois. The forum gives candidates an opportunity to introduce themselves and talk about the issues they feel are important. Prospective board members unable to attend the LISA conference will be able to submit a position paper to this forum. All candidates will be expected to respond for publication to a set of questions presented by the Nominating Committee. The nominating committee will coordinate these activities, and will contact each candidate before the election.

If you have questions about the nominating process, or what Board membership entails, please send mail to:<sage-nomcom@usenix.org> or contact a current member of the SAGE Board.

Moving Big Data Fast
by John Schimmel
<jes@sgi.com>

Most UNIX applications move data from disk into a process’s memory or from the memory of a process on one system to the memory of a process on a remote system. What that translates to, in administration terms, is disks, filesystems, and networks. To a certain extent, we can work only with what we’re given, but top-flight system administrators want to maximize whatever they can. This article examines the current state of high-performance I/O systems and tells how to pull the last drop of performance from your network and systems by understanding and exploiting the engineering that underlies them.

Fundamental technology has not really changed a great deal since disks were invented. Still, several factors have improved performance over the years: disks spin faster, so a track can be read in less time, the disks are much smaller, so the heads move much shorter distances; data are stored more densely on the media, so more data can be read on a disk rotation; and disks are more intelligent (with added buffers and hardware read-ahead to guess the next request). These account for nearly all of the performance improvements in disk drive technology in the last 20 years. Current drives can transfer around five megabytes of data in a second.

The communications channel from disks to memory has kept pace with the improved speed of disks. When deciding how many disks per communications channel, the rule to go by is to determine how fast a disk is, how fast the communications channel is, and never plug more disks into the channel than it can handle at peak disk transfer speeds.

Because the technology is not changing very rapidly, the secret to increased transfer rates is to use disks more effectively. The BSD filesystem proved that it was possible for an operating system to get nearly full disk performance. The optimization was to lay the data out very carefully on the disk to minimize head movement, keep the buffers full so that the disk was always doing useful work, and always to perform reads in multiples of the track size so that the most data are read for the effort. The work on filesystems in the last several years has been to update all the data structures for larger filesystems and add metadata logs to reduce the time spent in checking the disk when the system is booting.

There are times when the filesystem gets in the way (or, more often, the buffer cache gets in the way). Typical disk reads are for information that will be used by more than one process, so the operating system keeps a copy of these data in memory and gives processes a pointer to this buffer. The kernel does not know when a process is done with the data, so it is kept in memory until someone else needs the space. For data that are read off disk simply to test for the existence of a value, like a grep or a database search, processing large quantities of disk data pollutes the buffer cache; pages that probably will be used again get pushed out.

The buffer cache is one of the primary reasons that most database vendors build the database over a raw partition instead of on top of the UNIX filesystem. Several vendors now offer ways for the process to give hints to the kernel that these data are temporary so they should be copied directly into the process’s address space instead of through the buffer cache. On Irix this hint is given by using the O_DIRECT flag with the open call. Direct I/O requires extra programming insight because it usually bypasses some of the mechanisms that are used in the filesystem to get performance. If direct I/O is used, then reads should always be in multiples of the track size, etc. in order to get the same performance.

The key to extracting high performance from disks is in parallelism. Run as many disks as possible in parallel to increase the overall throughput of the media. Modern servers have put a lot of effort into the disk controllers so that they can perform DMA transfers straight from the disks into system memory with very little involvement by the CPU. On an
SGI Challenge machine, for instance, performance has been measured at over 500 MB/sec by stripping a filesystem over 128 disks.

High-speed disk arrays get high performance through this same use of redundancy paired with a big bank of memory to use as a cache. The bonus in using an array is that they typically implement hardware RAID 5. This scheme uses an extra disk to hold error correcting information so that if any one disk dies, not only are no data lost, but the data on the failed drive can be rebuilt when it is replaced. When using lots of disks to hold data for performance reasons, the chances of a disk failure increase. A concern with RAID is that calculating the error correction information is CPU intensive. Most modern operating systems offer a software RAID solution, but this slows the work of writing data to disk.

Like disk technology, network technology has not kept up with the increase in CPU speeds. The way to achieve higher performance over a network is similar to raising performance from disks. Increases in network performance are primarily from increases in clock rates (such as 100 megabit Ethernet standards) or increasing bandwidth from parallelism, such as the difference between the ATM standards or HiPPI. Although ATM is a reasonable telephony standard, it appears to be a rather poor networking technology. So the hope for raising performance in networking hardware falls upon increasing clock rates or increasing parallelism in those technologies already in use.

The current performance leader for desktops is 100 MB/sec Ethernet, with the gigabit Ethernet standard effort already under way. There is little doubt that the results of that effort will be the primary mechanism for moving data between desktop machines in the years to come. The 100 baseT standard is very similar to the 10 baseT standard with the clock rate increased and the timings changed a bit. The current question in the gigabit Ethernet efforts is whether the Ethernet packet format will change. At 1,000 MHz clock rates, it makes sense to move data in chunks much larger than 1,500 bytes, but changing this limit would require a revision to the Ethernet packet format. There is also some question as to whether the gigabit standard will be half-duplex or full-duplex, or perhaps both. The length limitations for cables will be about 15 meters, if things remain as they are now.

The high-end technologies of choice in recent years have been ATM OC23 and HiPPI, High Performance Parallel Interface. ATM is a very difficult technology to make work well and so has been slow to catch on. Most of the products that exist in the ATM space are on the low end, OC3 or OC12, and have been driven by hype. The HiPPI standard is very easy to understand and implement, and there are efforts under way to produce a very fast implementation that is more IP friendly than the current standard. Unfortunately there are many different HiPPI standards, including Serial HiPPI, which is somewhat of a misnomer. Basically, HiPPI is a point-to-point parallel DMA connection between machines. The latest work is on HiPPI-6400 (a.k.a. Super HiPPI), which transfers data at up to 800 MB/sec. This means that an 8 GB hard disk can be transferred over a wire in about ten seconds. Notice that this is significantly faster than current disk transfer speeds, however.

Network filesystems are getting faster and can now almost keep up with the midline networking standards such as 100 baseT. Building a network filesystem that acts like the local filesystem is very difficult because it must constantly be checking the state of the remote file to compare it with the state of the local buffer cache. The fact that NFS functions at all in a UNIX environment is really quite amazing! The way that NFS implementations have increased in speed is that most of them now cheat extensively. Version three of NFS has introduced asynchronous writes of data and caches of remote file attribute, as well as larger block sizes and a TCP implementation.

The way to transfer data quickly over a network is to write a lightweight file transfer protocol over TCP, such as FTP. Most vendor implementations of TCP are very highly optimized and correct. The SGI implementation of TCP, for instance, can get near wire performance over HiPPI, which is currently our fastest networking hardware. Again, there are popular tricks that have improved the performance of the implementation, but these have not reduced the correctness of the result. Version three of the NFS protocol is a move in this direction, as long as the TCP implementation is used.

The most popular tricks to increase TCP performance are hardware checksumming and copy avoidance. Hardware checksumming means that the network card checks the checksum for incoming packets and calculates the checksum of outgoing packets so that the CPU does not need to get involved. This was traditionally the most CPU-intensive portion of the TCP stack.

There are a number of methods used for copy avoidance. The most common is simply the careful layout of data in buffers as packets move up and down the networking stack so that the header information does not have to be copied between the layers. A more interesting method is page flipping, which is a popular UNIX trick. If a buffer for the read system call is aligned on a page boundary and is a multiple of the page size, then the user buffer is simply switched with the kernel buffer containing the incoming data by changing an entry in the page table. This avoids physically copying from kernel space into user space but has the same effect. Unfortunately, the page size in many kernels is now so large that this is useful only in very large transfers.
The direct I/O extension to the local filesystem is now being added to NFS with great success. When a file is opened with the `o_direct` flag, the data transfers do not go through the NFS and RPC code any longer, but instead go over a dedicated TCP connection to the remote system and are flipped into the processes address space. SGI is selling the results under the name of the Bulk Data Server, but it can be done easily in user space by directly opening the TCP connection between a server and knowledgeable user processes with similar results.

So, if you are in a position where you need to get as much data moving through your systems as possible, these are the things you need to do: Spread large numbers of disks over as many controllers as possible, get as many of these disks as you can spinning by striping data across them (being very careful to have the stripe size be a multiple of the system page size and the disk block size), and perform reads or writes that are in multiples of the stripe width (stripe size times number of data disks in the stripe). You should also upgrade to use NFS version three over TCP or (better) write your applications such that they bypass the remote filesystems (to exploit tricks the kernel can do that can be done only if you are reading and writing in large blocks). And you should be taking advantage of as many tricks that your operating system supports as is reasonable. You need to keep in mind that most of the tricks are very portable. Most vendors support a way to step around the kernel, but each of these is different.

The bleeding edge is the most interesting place to be, but it is also among the most painful. Keep in mind that specific acts done today will be out of date tomorrow, but the knowledge of how to make things go fast stays constant.

### Perl Practicum: Network Wiles (Part I)

*by Hal Pomeranz*

This month I begin a multipart article that examines network programming using Perl. Network programming with Perl is very much like network programming with C, but Perl's language constructs make it much easier to focus on the actual work of setting up a network connection, rather than issues like exception handling and data reformatting. People who have always been mystified by network applications can often use Perl to spin themselves up; veteran programmers can use Perl to prototype network applications rapidly.

```perl
use Socket;

$server = "www.netmarket.com";
$port = 80;

$server_addr = (gethostbyname($server))[4];
$server_struct = pack("S n a4 x8", AF_INET, $port, $server_addr);

$proto = (getprotobynumber('tcp'))[2];
socket(MYSOCK, PF_INET, SOCK_STREAM,
```

#### Thinking About Network Programming

Discussions of network programming always seem to devolve into mazes of twisty little acronyms, but the basic concepts are very simple. The easiest network relationship is between two hosts: a "server" that possesses some collection of data and a "client" that has a question it needs to ask the server to answer (hence "client-server computing"). For example, your Web browser is a client that can ask the Web server at my organization, “What information is stored at http://www.netmarket.com/?”

One useful analogy is those in-flight music systems that airlines have. The server is the airplane's music system; it has a bunch of data (movie soundtracks and different types of music) that it can supply to the passengers (the clients). The clients have to ask explicitly for the information, however, by plugging in a pair of headphones and dialing a little number to get the exact sounds they want to listen to.

The headphones in the above analogy stand for a concept that network programmers call a "socket." Clients establish socket connections to servers by connecting one end of a logical pipe to the server at a well-known address (the little hole in your airline seat) with a specific port number (dialing the number on your seat to get rock music) while holding onto the other end of the pipe (keeping the headphones on your ears).

As in my airline example, properly designed network servers can handle several client connections simultaneously. Unlike my analogy, network clients can connect to several different servers (or multiple times to the same server) simultaneously.

#### Doing It

Easily the most complicated part of setting up a socket is preparing the binary data structure that tells the operating system which server to connect to. This stuff doesn't look very Perl-like because we are preparing a C data structure—the Perl networking functions are directly tied back to the C socket library.

```perl
use Socket;

$server = "www.netmarket.com";
$port = 80;

$server_addr = (gethostbyname($server))[4];
$server_struct = pack("S n a4 x8", AF_INET, $port, $server_addr);

$proto = (getprotobynumber('tcp'))[2];
socket(MYSOCK, PF_INET, SOCK_STREAM,
```
$proto) ||
die "Failed to initialize socket: $!
";

connect(MYSOCK, $server_struc) ||
die "Failed to connect() to server: $!
";

The first line of this example simply pulls in the Perl sockets module. This module defines a number of useful constants that are employed later in the program. Next come the name of the server that this client will contact and the network port on which to talk to the server-port 80 happens to be the port that Web servers listen on, by default. You might actually get these values passed into your program as command line arguments, or this code might become part of a function that gets these values as function arguments.

In order to be able to connect to the server, the program has to translate the server's human readable name (www.netmarket.com) into a network address. The gethostbyname() function looks up the server name and returns a list of information: the network address of the server is the fifth value of the result (don't worry about the other values right now).

The C structure is created by using this address and the pack() function. This structure has three fields: a description of the type of network address in the rest of the structure, what port address to connect to, and what server address to connect to (the rest of the structure is just filled up with zeroes). AF_INET is a constant defined in the Perl socket module, which stands for an Internet Protocol (IP) type address (unfortunate people have to use other types of networks like AppleTalk, DECnet, or X.25, all of which have their own AF_* constants in Socket.pm). Unless the programmer specifies the type of network connection at the front of the structure, the operating system will not be able to interpret the network address information in the rest of the structure, and the attempt to set up the socket will fail.

With that messy pack() business out of the way, we can start setting up the actual socket. First, the client initializes its end of the socket as a Perl file handle, MYSOCK. The other arguments to the socket() function specify the type of network connection, how the socket will be used, and the transmission protocol. PF_INET is another constant from Socket.pm that is related to AF_INET and specifies that this socket will be an IP type socket (indeed, in the early days, AF_INET was used in both the C structure and in the socket() call—avoid and abhor this practice).

SOCK_STREAM is another constant which says that the client and server will talk using a connection similar to a telephone call—both parties can talk back and forth to each other and the connection will stay up until one party hangs up. (SOCK_STREAM is the most common communications method, but other methods exist such as SOCK_DGRAM which is more like smoke signalling—client and server can send out messages, but there is no guarantee that the other party will receive them.)

Finally, the transmission protocol is specified: the discussion of TCP versus UDP is beyond the scope of this article, but TCP is always the right thing to use unless you are very sure that it isn't. Always use getprotobyname() to get the right value for the TCP protocol number. Lazy programmers frequently hard-code this value because it happens to be the same on nearly every UNIX variant out there, and people like me curse them when I have to port the code to non-UNIX systems or strange UNIX variants.

With one end of the socket firmly in hand (again, as the file handle MYSOCK) the client calls connect() to actually contact the server. The connect() function takes as arguments the file handle and the C structure created earlier. Assuming the connect() succeeds, the client has actually established a session with the server.

**Using It**

MYSOCK can now be treated just like any Perl file handle, except that you can both read and write from the same socket. In order to save network and system resources, it is particularly important to remember to close() sockets when you are done with them.

Because this client has connected to the Web server (port 80, remember?) on www.netmarket.com, the client program can request an HTML document using the HTTP protocol:

```
select(MYSOCK);
$1 = 1;
select(STDOUT);

print MYSOCK "GET /\n\n";
while (<MYSOCK>) {
    print;
}
close(MYSOCK);
```

The first three lines turn off the standard I/O buffering on the socket. When reading and writing from a file, it is usually most efficient to do large reads or writes (read more data than needed or save up a lot of small writes and do them all at once), and most UNIX systems take care of doing this automatically. This behavior can, however, be disabled—for example, on a network socket where the client and server are passing short messages back and forth. The Perl mechanism for turning off buffering is to set the $1 variable to be non-zero (it's zero by default). Setting this variable affects only the currently selected() file handle (STDOUT is selected by default), so you have to select(MYSOCK), set the variable, and then go back to the default of STDOUT.
That done, the client requests a file from the Web server using the `GET` command in the HTTP protocol. The argument to `GET` is the name of the file requested (in this case, the client is asking for the file at the root of the document tree, but could just as easily have asked for: `/some/other/file.html`). The `GET` request is followed by two newlines.

Once the client makes its request, the server sends the contents of the requested file back down the socket (or an error message if the file was not found or some other error occurred). The standard HTTP protocol defines that when the server finishes sending the file, it hangs up its end of the connection—this causes the entire socket to be torn down. A client reading from a socket interprets this event just as if it had been reading from a file and reached the end-of-file marker. In the program above, the HTML document is simply being printed to the standard output.

**Practicing It**

The above example covers the basics of writing a network client program. There is a good deal of additional lore surrounding this subject, but there are a lot of people out there earning huge salaries who don’t know anything more than what you have seen here. In the next issue I will explore server programming by writing a simple Web server.

In the meantime, practice these concepts by taking the example above and writing a program that will take the server name, port number (default to port 80), and file name as command line arguments and fetch that file from the remote Web server. Impress your friends (and increase your productivity) by building a Web robot that surfs the Web for you by looking for `href` tags in the documents you download and then fetches those documents as well (making sure that you don’t download the same document twice!). Now make sure the robot stops at some point, or you’ll download the entire Web.

**From the SAGE Secretary**

_by Timothy Gassaway_  
<gassaway@usenix.org>  

In a move to provide members with a steady stream of SAGE information, SAGE Board secretary Tim Gassaway will provide monthly member updates. The planned updates will include hot new SAGE issues and updates from the SAGE Board on current and projected projects. All members are urged to send comments and pass along information and issues that would be of interest to other SAGE members to <gassaway@usenix.org>.

**Hall Talk at the SANS Conference**

The USENIX/SAGE booth had a flurry of activity all week. The SAGE salary survey really generated the traffic: the summaries will be featured in an upcoming issue of ;login:. We added 29 new SAGE and 30 new USENIX memberships from walk-up sales and many more from the SANS conference registration.

We met many new and old friends while at SANS, and from the dialog the following suggestions were generated:

The “SAGE news of the month” will be posted to `sage-members` and the Web page.

The idea of “MasterClass” tutorials for higher level sysadmins evolved. The MasterClass tutorials would assume a basic knowledge of the topics at hand. “MasterClass: sendmail,” for example, would jump right in and might deal with rewriting rules, tuning, etc. Some other suggested topics were:

- large-scale network design  
- best practice routing topologies (including OSPF)  
- enterprisewide email architecture  
- enterprise change management (software)

**SANS BOF: dc.sage**

The dc.sage gathering held a PGP key signing party.

A suggestion was made that local groups write a column in ;login: to let other local groups know what they’re doing and perhaps work together.

The dc.sage group would like SAGE to develop a speakers bureau for local groups to draw on to obtain qualified speakers for their meetings.

The question was asked if conference tutorials could be referenced back to the SAGE Job Descriptions handbook. A request for more advance tutorials was also mentioned.

Carolyn J. Sienkiewicz <cjs@mrj.com> and Brad Knowles <bknowles@aol.net> are the new dc.sage contacts.

[Author’s note: dc.sage really impressed me as being a great group of people with like interests who truly care about their peers in system administration. The frequency of meeting times, dates, and places seem to be adaptable to meet the needs of their members. dc.sage, keep up the good work!]

AUGUST 1996 ;login: 23
SANS: The System Administration, Networking and Security Conference Reports

Washington, DC, May 12-19, 1996

Sponsoring organizations: SAGE, The System Administrators Guild; The Network Security Institute; The Web Development Institute; and FedUNIX.

SANS Tutorials

Summarized by Idajean M. Fisher
ides@psa.pencom.com

UNIX Security: Threats and Solutions
Matt Bishop, University of California, Davis

A practical overview of a wide variety of issues surrounding known UNIX system security threats, this was an excellent choice for anyone involved in the planning, design, or implementation of a contemporary UNIX-based IS effort. Strong emphasis was placed on case studies, which gave the course a good sense of balance and perspective.

Course work started off with a look at password security, including the mechanisms by which passwords are generated, stored, shadowed, and potentially cracked. A detailed description of the workings of the UNIX password hashing mechanism was particularly worthwhile, as was a discussion on a variety of publicly available tools to help scrutinize password robustness and a discussion of pros and cons of various password aging mechanisms.

Also covered were strategies and mechanisms for effective file and file system auditing, including file encryption techniques, value of file checksums, dangers inherent in having unidentified SUID and SGID code, gaining root access by exploiting race and buffer overflow conditions, security considerations for NFS mounted file systems, and storing and protecting system logging and accounting information. There was also a short but well done section on incident detection, analysis and management after the fact, with an emphasis on dealing with “human factors.”

Effective Security Incident Response

Gene Schultz, SRI International

This tutorial presented a 50,000-foot overview of issues surrounding productive incident response, including considerations for developing a useful response procedure to deal with security-related issues before they occur. Concepts touched on included how to deal with management effectively, how to communicate in a positive fashion with other sites that may also be affected by the incursion, and how to avoid tipping off the attackers that you’re aware of their activity in the process.
A significant amount of time was spent discussing the legal ramifications of incident response. All in all, this was a good tutorial for nontechnical managers with significant responsibility for security issues, incident management, and the establishment of confinement/control practices.

SANS Technical Conference

*Summarized by Idajean M. Fisher
ides@psa.pencom.com*

**Session: From Rags to Riches: Redefining the Role of Today's System Administrator**
Ed Taylor, Pencom System Administration

This discussion of trends in the open systems industry as a whole and their impact on the changing role of systems administrators placed strong emphasis on the shift from thinking of sysadmins as failed programmers to the most sought after people in open systems today. Significant empirical evidence was presented to support this position, including manufacturer statistics on how many physical machines were sold in the US last year versus the number of currently existing sysadmins. Data presented suggested that some 1.8 million new UNIX boxes were sold last year. The assumption was made that a particular sysadmin can handle 100 of these systems. In light of these numbers, it was suggested that to manage the systems sold last year our industry requires in the range of 18,000 new sysadmins. Combine this with the estimate that there are only about 20,000 UNIX sysadmins in this country as a whole, and the numbers presented were a pretty good indication of the magnitude of the shortage of talent available and how valuable professional sysadmins are to industry!

Estimates for the number of new NT installations were even more amazing. It is predicted that this year over 4.5 million NT boxes will hit the market, but Microsoft anticipates being able to train only some 4,500 administrators during the same time period. All this seems to bode well for professional system administrators—an excellent argument to take a hard look at current employment options and alternatives.

**Session: Commercializing an FTP Service**
Mike Fuller and John Stewart, Cisco Systems Inc.

This session described the evolution of a large-scale anonymous FTP server as exemplified by Cisco Systems' current service offering the CIO (Cisco Information On-line). The talk outlined the migration of a basic FTP server implementation through to a relatively complex, high-volume system with multiple service offerings to support Cisco Systems' customer base. Over the life of the project, they have had to deal with a variety of issues concerning scaling, ease of use, effective search mechanisms, and revision/source control. How the authors were able to use a modified version of WU-FTP with a reasonably advanced access control mechanism, strict publishing requirements, and a variety of usability enhancements to strengthen, enhance, and expand their information service was described.

**Session: Security Issues with Mobile Computing**
Dan Geer, OpenMarket

This presentation was a high-level look at security issues in an ever changing computing environment: trying to understand how the concept of a “Trusted Computing Base” maps to a mobile-based IS organization where connection and service requests are made from geographically varied and difficult to track locations; possible impacts on conventional security planning and new considerations relating to physical security in an increasingly mobile computing environment; authentication issues in a mobile environment (how to prove that a moving target is who it says it is); implications for conventional authentication methods as well as for emerging technologies; and migration toward onetime password mechanisms and possibly biometric-based authentication methods.

**Session: Detection and Prevention of the Electronic Intrusion**
Alexander Yuriev, Temple University

This coverage of trade-offs and sanity-based planning for security intrusions looked at “when” as opposed to “if” security incursions will occur and the relative appropriateness of reactive versus pro-active approaches to system security. Specific examples were given for the following pertinent topics:

- Some ways to detect unfriendly probes by attackers looking for system vulnerabilities
- Detecting attackers using a system for a Jump Station or platform to attack other systems
- Attempts on the part of attackers to cover their tracks by altering or destroying system or auditing logs
- Uncovering back doors left by an attacker to regain access to the system for subsequent use

**Session: The Future of the Internet**
Rob Kolstad, BSDI

This engaging, useful, and highly interactive presentation on the future of the Internet highlighted some of the often unrealistic expectations of people and organizations trying to cash in on what is currently perceived as an Internet gold mine out on the horizon. There was provocative discussion on the reality of growing profitable Internet-based service
businesses and some of the profitability issues and concerns surrounding getting a new ISP off the ground. A tentative look at the future of phone over the Internet, included feasibility, a possible working model, and obstacles. Also discussed was a series of level sets for what the true target market for these future Internet service offerings might be, including an attempt at determining their relative comfort level on the “Mrs. Kolstad” (Rob’s mom) comfort scale.

Session: Firewalls Are Dinosaurs: The Great Debate

Marcus Ranum set the pace for the debate by taking the part of the “defense” on behalf of firewalls. All in all he made some persuasive arguments, but to be honest I thought his use of “if it doesn’t fit, you must acquit” a little cheesy. There were lots of car and highway analogies, and for the most part panel members seemed to agree with each other at the top of their lungs while violently claiming to defend opposing positions. It was a lot of fun to listen to, but at the finish it was unclear (to me at least) who won “The Great Debate.”

SANS BOF: WISA: Women In System Administration.
Moderator: Idajean M. Fisher

This was a discussion of various issues concerning gender equity and a variety of other points that affect and concern Women In System Administration today. Although the group was quite small, the conversation was productive. Some of the issues discussed were pay equity, socializing with professional peers in a predominantly male environment, the second woman effect, and feelings about the quantity and placement of women seen when interviewing for a new job— all in all a pleasant discussion. It was also nice to see people from last year’s WISA at LISA BOF.

SANS Technical Conference

Summarized by Pat Wilson
paw@rigel.dartmouth.edu

Session: Current Trends in Intruder Methods
Moira West-Brown, CERT

Moira West-Brown gave an interesting talk about CERT’s experience with security incidents and the trends they’ve observed. Starting with a background of the Computer Emergency Response Team (formed in response to the Morris worm), CERT became a clearinghouse for reports of security breaches. This was initially a manageable task, but the growth of the Internet and the number of such attacks have caused CERT to enact a strict prioritization of effort. Its top priority is incidents that are “life-threatening” or that jeopardize Internet infrastructure (attack a host, and you’ve annoyed tens or at best hundreds of people; attack infrastructure routers or root nameservers, and you’ve affected tens or hundreds of thousands of people). Other national CERT agencies exist to track matters in other countries.

CERT has counted 2,412 incidents (so far this year), where “incident” is loosely defined. Though the number of incidents continues to grow, the rate of growth of reports to CERT has slowed (no mention was made that CERT’s practices might have had some effect on this). The most common attacks are (1) gaining Internet access—it used to be attacks on dialup terminal servers; now attackers provide bogus credit card numbers to ISPs. (The card number, a simple checksum, isn’t actually checked for up to 24 hours.) (2) attacks on poorly secured “newbie sites,” using ISS, etc., to probe for holes.

Once inside, attackers may install trojan versions of programs and/or leave a shell running on a high numbered port. CERT has noticed an increasing sophistication of attacks (first there were trojaned logins, then trojaned Telnet, and now packet sniffers are common). It’s obvious that source code is being read to get a good exploit.

Sniffers and sniffer tool kits (e.g., rootkit) seem to have become big in 1994. An NFS tool kit (the non-exploit version of which is available from the COAST archives) appeared as well. The year of IP spoofing was 1995 (170+ infrastructure attacks). CERT saw an average of three spoofing attacks per week on CERT systems. Attacks that hijacked open terminal sessions were seen. Exploitable holes in httpd were discovered.

So far, 1996 seems to be the year of Java holes.

In response to a question from the audience, Moira stressed that vendors aren’t providing better security because the customers aren’t demanding it—security constraints aren’t written into RFPs, and security concerns aren’t an obvious part of “buy decisions.” [Author: Otherwise, why would companies buy machines like SGIs, which are insecure out of the box?]

Because of the increasing demand for their services, CERT had initiated a “CERT Affiliates Program,” which offers seminars, training, and “priority access” to CERT services (essentially, pay them, and they might talk to you). Fees appear to start at $25,000/year. [Author’s editorial musing: are they pricing themselves out of the market?] Moira stresses that this doesn’t mean CERT won’t still collect information and (slowly) disseminate it, just that affiliates will get priority assistance. [Author’s editorial musing: and, at $25K+, they’ve bought it.]
Session: Adventures in Hackery
Matt Bishop, University of California, Davis

Matt gave an entertaining talk, illustrating security “truisms” with examples from real exploits (or, in some cases, exploits he’d like to see).

The cardinal rules of securing systems are striking the correct balance (the principle of “psychological acceptability”—if you annoy users with too stringent security requirements, they’ll find a way to go around them, and your security is shot) and “knowing your assumptions” (illustrated with an anecdote about a misconfigured Secure RPC implementation wherein bogus updates to NIS maps could be made because of unfortunate fallback behavior of the Secure RPC system). The wrong assumptions can create large security problems.

It was pointed out that most firewalls are easy to defeat from within—one “backdoor” connection around the firewall (or modem connected to a machine within the perimeter) can destroy the security the firewall was meant to create. It’s also vital to keep up with announced security patches (though one must understand what the patch is doing, lest it compromise security in another way or interact badly with other software on the system).

Some attacks rely on poor assumptions—an example illustrated why restricted shell accounts should never rely on the fact that the root account is “trapped” within a chroot for security, and hence SUID/SGID programs should never be placed within restricted shell accounts.

Matt reiterated that technological controls (sniffers, watcher programs, etc.) should be used when possible and that, in security, if there’s an easy fix to a potential problem, it should be fixed.

Dear Miss Management

[Editor’s Note: A BOF session at the recent SANS conference exposed me—for the first time—to the fact that a number of administrators have management that is actually the target of venom and even hate! We’re talking Dilbert incarnate—incapacity beyond belief.

Being a big fan of USENIX as “facilitating communication,” I have recruited another columnist to write about management. Since it is a fairly touchy subject, this columnist is remaining anonymous for the time being. Please address your questions for Miss Management to login@usenix.org and I’ll forward them. RK]

[From Miss Management: This column is gleaned from a discussion at SANS.]

Dear Miss Management:

I just can’t figure out what’s wrong with my manager. We have a medium-size shop with lots of NT machines and a single UNIX-based mail server. That server is the pulse of our entire organization and must be 100% reliable and functional. It is inconceivable that the application can run on NT. I have explained to them (with charts and graphs!) all the technical reasons why UNIX is really the only answer and yet they stubbornly insist that I must change my mail server to an NT machine. I am at the end of my rope. What shall I do?

Signed: Really, really, really frustrated

Dear Frustrated:

Of course you are frustrated! Frustration centers on expectations that are not met. You expected a rational presentation with technical facts and explanations would convince your management to make the “right” decision. They did not do so and you do not know how to help them. That is the very recipe for frustration.

Now, what to do?

First of all, the lack of words like “stupid” and “ignorant” in your note indicates that your relationship with your management still has at least a shred of mutual respect (and can presumably foster a bit of teamwork). This is good.

I often find it instructive to put myself on the other side of the problem and consider their point of view. Let’s try that.

We have a manager with, say, 55 NT machines and a lone UNIX box. Being a big picture person, the manager knows that UNIX is incredibly difficult to administer. While this might or might not be true, it is certainly common knowledge throughout the trade magazines.

Furthermore, we have Microsoft marketing and sales efforts which are so compelling that many managers are willing to drop everything just so they can have NT running at their site. The notion of a technical “problem” or “deficiency” just isn’t the issue—Microsoft will have a solution shortly if they don’t have one now.

Given this mindset (and, despite the fact that it’s wrong, it does make sense), what decision should a manager make?
Of course we need to get off the UNIX box! We simply can't afford to administer a completely parallel environment! We must standardize; everyone knows that!

Note that by simply changing the focus of the decision-making process we have come up with at least one (maybe the right one, maybe not) way for management to view the problem and make decisions.

You need to learn what is motivating your manager in this decision making process. Is it budget? Does he/she golf with a VP from Microsoft? Is it a directive from above? Is it fear of the unknown? I am fairly sure the issues are not technical. Even if they were, I sense from your letter that your manager is not a highly technical person.

So, generally, here is the key: Just because you know the answer to a technical issue doesn't mean your management can hear you. Find out what their evaluation criteria are, and pitch your solution using their criteria, not yours.

Good luck!

A Different Kind of Networking
by Barbara L. Dijker
<barb.dijker@labyrinth.com>

I promise. This article is not about the Internet. We're all a little sick of hearing about the Internet. I know I am.

You've probably heard people in suits talk about "networking," and you know they've never touched a keyboard. You may have caught a glimpse of the term in business seminar brochures. But you probably figured that because it has nothing to do with computers, it's not for you.

A network is a means to share information and resources. We use our computer networks to share information and resources all the time. Isn't that sufficient? Isn't everything I need on Archie, Yahoo, or USENET? The answer to that is a resounding "NO!" Interpersonal networks depend upon building relationships with other people—you know, humans: ugly bags of mostly water. They provide access to information and resources that you may not otherwise have without the relationship as the medium. An interpersonal network is the network of friends, colleagues, co-workers, and acquaintances with whom you have a good reciprocal relationship.

How do you start building one of these "networks"? It's easy actually. The quickest way is to begin or join a local peer group. This can be anything from going out to lunch with your co-workers regularly to joining/forming an organization that has a board of directors, a budget, and regular public meetings. Let's start simply.

To join an existing peer group, you have to know it exists. Ask around: there's bound to be something. Otherwise, you're elected to start one.

Start with an internal company peer group. Meet on a regular basis for a common activity (like food). You don't have to do anything except eat and swap war stories. If your company is small or you want to expand your "network," include peers from other companies in your local area. Create a mailing list for the group. The group can be as exclusive or as open as you like. An exclusive group allows the members to be more comfortable in exchanging their ideas and information if they are personally acquainted with the other members. A more open list gives you more breadth. But as a rule of thumb, everyone in the group should know at least one other member on a personal basis. This shouldn't be difficult, and it provides a path for introductions for the others.

In my area, there are a few such groups. One has been around for probably ten or more years. It's rather exclusive—by invitation only, more or less. In fact, at one time, the running joke was that to be in the group, you had to have "slept with" a founding member—where "slept with" was very broadly interpreted, e.g., in the same house, USENIX conference hotel room, on a road trip, camping trip, etc. So meeting this requirement wasn't as difficult or unseemly as it may sound. Many of the members of the group were system administrators full-time or part-time at one point in their lives. Today, they range from professional system administrators to kernel hackers and numerical analysts. Members now span two to four continents, depending on the time of year. There is a mailing list and they meet for breakfast once a week. A computer program decides where they go each week and sends out announcements (true to geek form). Originally, they met for lunch, but once the workday is started, it's tough to escape. So breakfast it was, and 8:00 am at that. Of course, not all of them make it to breakfast. The mailing list is enough.

What good is such a loose association? On this local mailing list, the predominant messages are jokes and personal messages like wedding or baby announcements and "so-and-so is in town, let's go drink." There are occasionally postings for jobs: have or want. There are also postings of security events or "in case you didn't know—I just beat this code into submission and this is how." The group provides a sounding board.

The point is that these are people you know personally, who know you, are in the same industry, and whose opinions you respect. Through this network infrastructure you can gain or provide (remember, it's a two-way street) information and resources. As the network grows, matures, and evolves, it becomes more valuable. Maybe you urgently need to borrow an old QIC cartridge tape drive for an afternoon. You won't readily find that on the net. Maybe you are considering hiring someone from a nearby company. If someone from that
company is in your group, you may get a more candid assessment of your candidate.

If you are particularly energetic, you might want to start a more formal and public local peer group. This takes time to organize. But don’t let the potential for bureaucracy intimidate you. A local group may be formed without going through the process of legal incorporation. All you need is one or more people willing to coordinate meetings, a place to meet regularly, and a means to get announcements out.

The trick to a formal group is to avoid money if you can. Once you collect money, you have to deal with a legal existence and a bank account. Many universities, government centers, or companies have large meeting spaces they can make available for at worst a nominal fee. You can get local companies to sponsor meetings: paying for the use of or providing the meeting room and copying and/or mailing the agenda or flyers in exchange for an advertisement on the back. Then all you need are the ideas for meetings. Draw from talent within your group and local companies to make presentations or lead discussions. If you start a public group, no matter how informal, notify the sage-locals working group.

Whether you get involved in a small informal group, a larger formal group, or multiple groups, you’ll be expanding your contacts in the field. The old saying rings true: “it’s not what you know, but who you know.” Worst case, it can’t hurt, and you might make a few friends in the process.

How Local Groups Can Make a Difference

by Carolyn J. Sienkiewicz <cjs@mrj.com>

The Washington, DC metropolitan area local SAGE group, aka “dc.sage” has been an important resource to me as a systems manager. If there isn’t currently a local group in your area, I encourage you to make an effort to start one. If you’re not sure the effort is worthwhile, read on and maybe I can persuade you.

Each of us is busy in our own little corner of the technical world. It is impossible for any one person to keep up with all of the new technologies, not to mention the political and legal influences increasingly coming into play in our profession. Staying in touch with dc.sagers through our mailing list, and attending the monthly meetings is a way to help bring it all together for me. This is the one place where I can regularly get in touch with a wide variety of peers.

The impact of blending our local information is best shown in the area of jobs. Many of our members have gotten new jobs or co-workers because of our job listings. For example, I filled a position vacated by the dc.sage founder, Ken Mayer. Exploring jobs with other local group members provides information, and allows for a better decision about whether a position is right for you. Similarly, if you hire someone from the membership, you’ve had a chance to speak with this individual several times over the course of months and really get a feel for their capabilities and personality.

dc.sage members are currently engaged in a local ISP quality study. Many are cooperating to gather statistics on the service they are receiving from their local ISPs. It is hoped that the results of the study can be valuable to all members, first by showing which ISPs provide the best service, and second, by providing members leverage with which to demand improvements.

For the newer sysadmins, local group meetings can be like striking oil. For the more seasoned veterans, meetings provide an opportunity to pass along knowledge and experience and to contribute to the teaching of the ever new crop.

But ultimately, the best feature of these meetings is being among your own kind. Personally, I feel a very special bond with the people I’ve met through dc.sage. We offer each other hard to find friendship, support, and a shoulder to cry on. These colleagues understand how the experiences of a sysadmin can make you feel. They have been there too.

So, when we get together for our “Sysadmin Summer Olympiad ’96,” it won’t just be to win the coveted gold for “Pin the Tail on the Luser.” And it won’t just be to flex those muscles and get out those hardware aggressions in the “Hard Disk Shot-Put” event. It will be to celebrate the community that each of us has helped to create. I am very grateful for the contributions, involvement and generosity of each and every member.

SAGE Award Nominations

SAGE is soliciting nominations for its fourth annual Outstanding Achievement Award, to be presented this October in Chicago at the 10th Systems Administration Conference (LISA ’96). The SAGE board has set up a special committee to select this year’s recipient, and we’re inviting your suggestions. The awards committee would like to keep the selection process informal; there isn’t a formal nominating procedure, and we will consider all suggestions submitted. So please send in suggestions for people whose professional accomplishments you believe deserve the recognition of a SAGE Outstanding Achievement Award to <sage-award@usenix.org>.
Qualified consultants are poised to leverage their expertise in the flourishing Internet market. What it takes to be successful at it is an increasingly popular topic at systems administration gatherings. This issue, SAGE will focus on the realities of consulting, from perceptions to financial equations.

SAGE hopes to focus on other topics of interest to the systems administration community. If you have suggestions for these topics, please let Tina Darmohray, <tmd@usenix.org> hear from you.

SAGE FOCUS: BEING A CONSULTANT

Editorial
by Tina Darmohray, SAGE News Editor
<tmd@iwi.iwi.com>

I recently formed an investment club with some of my friends. We pool a certain amount of money and invest it in the stock market; we share the profits (or losses). So nowadays I tune into “all-news radio” to hear the twice hourly financial reports. I always gain new insights from the financial reporter, but I sure hate the commercials. Full of get-rich-quick schemes and ridiculous products, they seem to be an order of magnitude worse than other stations’ commercials. I can’t believe that anyone takes them seriously.

The other day, though, one of them hit me close to home. I was startled and offended at the thought of just how misleading the advertisement really was. I was also concerned that someone might actually believe it! This ad claimed their organization could give you all the information you would need to be an “Internet Consultant.” It went on to paint colorful analogies of the Internet as a modern day gold mine—and those who exploited it as the “forty niners.” It assured listeners that “you really don’t have to know that much to make money” as an Internet consultant.

It’s that last line that I take issue with. Although the strictly mathematical analysis of that sentence is true, i.e., someone could make money without knowing very much, in the context of the commercial, the message is that anyone can be a wildly successful Internet consultant with little or no previous experience. That’s not realistic because long-term success as a consultant depends a lot on your reputation. You need to be doing a good job, or you’re not going to be in business very long. Real success isn’t just making money once: it’s continuing to make money.

Being a consultant specializing in Internet applications is a great career choice right now, but it’s also a demanding one—if you are truly successful at it. You need vast experience so that you can hit the ground running in a variety of customer sites. You have to have depth as well as breadth of knowledge. Clients often need a specialist, someone who has a particular expertise that is hard to come by. You need vision and willingness to work hard. Not only do you have to recognize and fit the solution into the big picture, but at the end of the billing period you have to have delivered something their staff wasn’t able to without your help. And you have to know about business. You need to be able to market your skills, manage your time and resources, invoice your customers, and pay Uncle Sam (and a host of other entities that lay claim to a piece of your successful business).

That’s a lot of things to “know” that the advertisement left out; I’m sure that’s what they’re going to cover when you pay them to tell you about it . . .

I still think that misleading ad was almost shameful, and anyone who is inclined to rush out and quit their day job is going to be disappointed when reality sets in. But don’t just take it from me! I’ve asked some other consultants to share their opinions with you. If you’re considering taking the plunge to the ranks of the self-employed, here are some words of wisdom from individuals who can claim sustained income from it. These might let you know if it’s “for you” and how to make it work if you decide it is.
So you wanna be a consultant...  
I got my first consulting job in 1984. My ex-employers decided they were wrong in shutting down the UNIX project and laying us all off, so they restarted it. Because none of us was dumb enough to go back to work there again, I was asked to bid on a project to train people in UNIX and C. They wanted a flat-price bid, so I figured out how many hours it would take and bid it assuming 1.5 times my old hourly rate (hey, it was like overtime). Needless to say, I got the bid. The project came in on time, right on the budgeted hours, and they paid promptly.

I didn’t make anywhere near 1.5 times my hourly rate on the project.

I made just about every financial mistake a new consultant could make. Since then, a lot of time and experience have gone by. I’m now a full-time consultant with a collection of regular clients who pay a lot more than 1.5 times my old hourly rate. This article talks about some of the lessons learned along the way.

As I write this, it’s a few days until June 17, when I have to pay my quarterly taxes. If you’ve never had the joy of doing this, imagine having to go through all the normal IRS tax stuff five times a year. Four times a year you must make estimated tax payments that approximate what your employer would have deducted from your salary had you been employed. The fifth time is when you have to do your taxes for real (and no, timing won’t permit you to merge the fourth estimated payment with your real annual return).

It’s been my unpleasant experience that people get “good-paying” consulting jobs, then find that they’re making less money than they did as a regular salaried employee. This comes from several common errors.

To make the example work, we’ll assume that you’re currently in a salaried position at $50,000 per year, and your goal is to make $100,000 per year as a consultant. We’ll assume you’re currently paying $7,000 in income taxes and $3,000 in Social Security, so you’re clearing $40,000 a year. I’ve deliberately picked simple numbers to make the math easy, but the numbers we get below are reasonably accurate. (But your mileage may vary, a point to which I’ll return.)

Real Income Taxes

In this field, almost everyone is close to or at the highest tax bracket, 28%. Because income taxes are graduated, the average tax on your overall income is much less than your incremental tax. Incremental tax is the amount of tax you pay on the next dollar you make.

So let’s say you get a consulting job and earn $100,000 per year. Your income taxes are now $7,000 plus 28% of the additional $50,000, or a total of $21,000. Your income doubled, but your tax bill tripled. It doesn’t much matter if you like this or not—it’s the way progressive taxation works, and that’s what we have here in the US and many other countries. Feel free to complain, but do it to your congressperson rather than me.

Federal Social Security

If you look carefully at your pay stub, you’ll find a deduction for Social Security. At the current rate of 6.2%, you’re paying around $3,000. What you don’t see on your check stub is the matching portion your employer has paid. When you become a consultant, you are considered to be your own employer and are responsible for both halves, or 12.4% of your gross. The calculation gets more complex, though, because you pay only on about the first $66,000 of income. So your Social Security taxes are about $8,000. There’s also a 2.9% Medicare tax that you pay on all income, $2,900 on $100,000 gross.

The bottom line: you’re paying $21,000 income tax, $8000 Social Security, and $2,900 Medicare. This comes to $31,900, bringing your take-home pay to $68,100. In exchange for doubling your gross, you get only a 70% increase in take-home pay. But don’t get happy yet; things are going to get worse.

Real Social Security

Anybody in this field who expects to retire on Social Security and maintain his or her current life-style is nuts. You may be fat, dumb, and happy at 30, but 20 years from now you’re going to be fat, smart, and pretty damned nervous about retirement.

If you’ve got a real job, your pension will likely be much better retirement than Social Security. But if you’re self-employed, there’s no pension waiting for you. Fortunately, the federal government has made some very nice provisions allowing the self-employed to build pension plans for themselves. Keogh plans, IRAs, and others are easy to set up. See your favorite accountant for some recommendations of financial planners, set one up, and fund it.
The government allows you to put up to 15% of your pretax income into a retirement account. Because of some various jiggery-pokery, it actually winds up being only a shade over 13%. If you’re at all serious about consulting as a career, you should be contributing the maximum, especially if you’re young and healthy. Compound interest is your friend, and if you don’t start until you’re 40 it’s damned near too late. So get an IRA or Keogh, and put 13% of your $100,000 into it.

Congratulations. Your $68,100 just became $55,100. And your 70% take-home increase is now 37%. But you’re probably going to do much better than the corporate pension plan. For one thing, you vest instantly. For another, it moves with you from place to place and just keeps growing.

Real Insurance

If you’ve not priced health insurance, get ready for a shock. Single young adults can get by pretty cheap, but a couple at 40 can easily spend $5,000 per year for good insurance. And remember, for whatever medical insurance you don’t get, you’ll have to pay actual expenses out of your pocket. Dentists aren’t really horribly expensive for adults with good teeth, but costs can add up (let’s not think about kids with braces).

But that’s not all you need. Whether you have a family or not, you need to think about disability insurance. As a consultant, you get nothing from workman’s compensation or long-term disability through your employer. My insurance agent says that one person in five goes on extended disability for some period of their lives. If you’re a young single person who can live with parents if a disaster happens, you might want to risk it. But if you’re the primary earner for a family, it’s not a smart risk.

If you have a family, you’ll want to consider some additional term insurance and some whole life. You can get incredibly good deals through IEEE or the ACM, but it’s still an expenditure.

Obviously, costs for this level of security vary wildly, depending on your age and circumstances. But for the purposes of our discussion, let’s assume your annual insurance plus new medical costs are $7,100. This takes the $55,100 down to $48,000.

Congratulations! Your 100% increase in gross salary is now a 17.5% increase in your take-home pay.

When A Year Is Not A Year

Another fundamental mistake you can make in consulting is assuming you will remain employed 40 hours per week indefinitely. There are those who do, but at the beginning, you will find yourself in one of two situations.

If you take a full-time contract with a large client, you’ve effectively removed yourself from the job market. When the contract is up, you’re going to be unemployed. Because you’ve been out of the market for a while, it’s going to take some time to get a new placement.

It’s also possible to work for a large number of small clients on retainer-type bases, so that as clients come and go, the demand stays fairly steady. But it will take a while to build that stable of clients, and in the first few years, you can go through some really lean times.

The upshot of this is that either way you shouldn’t assume you’re going to work 40 hours a week all year. In the beginning, assume it’ll be more like 30 hours a week, or 1,500 hours per year. Later, you can adjust based on what happens in your real life, but you’re never going to get to 40. You still want vacations, holidays, and those week-long trips to USENIX and LISA. It’s a safe bet that you’ll use about five weeks a year on those items, so assume you’ll max out at 36 hours a week.

Now the math is easy. At 30 hours per week 52 weeks per year, you have to charge $65.00/hr to make $100,000—and remember, you’re still only taking home 17.5% more than you were before.

Beyond this point things start to vary wildly, depending on your circumstances. What I want you to be aware of with the above example is that there are bears in the financial woods, and some of them are damned expensive bears.

Expenses

One nice thing about being a consultant in this field is the low overhead. For day-to-day work you’ll need to buy a computer, modem, printer, phone line, probably an Internet service, and a room in your home to put it all in. These aren’t big numbers, but assuming you replace the computer every few years, it will still come to between $1,000 and $2,000 per year. These expenses come “off the top,” i.e., they reduce your taxable income, so they don’t hit the bottom line as hard as IRAs and such.

You’ll also want some good accounting. I swear by (not at) QuickBooks, but that’s not all you need. Good tax accountants save you far more than they charge and give much more accurate advice than your Uncle Joe (or me) about what is really allowed and what isn’t. Trust me, it’s money well spent.

You’re also going to need to keep current in your field. That means books, conferences, and maybe training. If you don’t live nearby, your cost to attend LISA or USENIX and take one tutorial will be around $1,250. Again, it’s all deductible, but it still adds up.
You’ll also probably spend more on books and documents than you used to. Assume at least $500; my spending is more like $2,000.

Our example consultant is now spending $3,000 to $4,000 per year on expenses—not a lot, but even after tax savings, it still takes $2,000 to $3,000 off the bottom line.

**Doing Your Planning**

If you are considering becoming an independent consultant, you need to go through an exercise similar to the above. The first step is to decide how much you’d have to take home in order for consulting to be worthwhile. This gives you a target number to shoot for. Then sit down with a spreadsheet that reflects your taxes and expenses, and start plugging in numbers.

At some point, you’ll answer the question of how much you must charge in order to make consulting worthwhile for you. If you’ve got customers ready who’ll give you the right number of hours at the right rate, great! Welcome to the biz. If not, you’ve got some tough decisions to make. Is your problem coming up with the client base, or is it that the clients you have aren’t willing to pay what you need? And don’t just look at the numbers you need to make your minimums. Look at what happens if you miss by 10% or even by 20%. You might need to stick to moonlighting for a few years until your skill set and client base are bigger, or until you’ve got enough savings to endure some wild income swings. Or it could simply be that consulting isn’t for you.

**A Real-World Example**

The point of this article is not to scare you away from consulting. It’s to make sure you’ve really thought through some of the issues. My own situation is somewhat different, and it’s worth looking at as a less pessimistic example.

For the past three years, the total amount I’ve paid in taxes, Social Security, and IRA funding comes to 36.8% of taxable income—considerably better than the numbers shown here. But we are homeowners, and that alone makes a huge difference in income tax.

Most of our medical insurance comes through my wife’s employer, but not all of it. When I was employed full-time, my employer covered the rest. Now we must cover that additional cost ourselves. We’ve also got two children, which drives up the medical costs. All together we spend about $3,000 on additional health, life, and disability insurance.

Taking all the additional costs into account, we determined the minimum gross income needed to hold steady. Assuming I could bill 1,500 hours per year, it came to $50 per hour. But that wasn’t good enough. If we were going to assume the additional risk and worry of having one of us self-employed, it had better be worth it financially. Working that through, it came to $60 per hour, or about 2.3 times what I was making hourly before the change. Once I had the client base to support it, I said goodbye to my day job and have only rarely looked back.

These days we make our goal numbers easily. To get stability, I try to keep any single client from becoming more than a third of my income. Whenever the billable hours creep near 1,700, I raise the rates. This always makes a client or two drop off the bottom, but the increased rate makes up much of the difference. Over the course of a year, the lost clients are usually replaced, so for the last three years, I’ve had a $5 to $10 increase every year.

So we’re now doing quite well, thank you. I’m able to do a few non-paying things just for fun–like writing this article. But it took eight years to move from the first consulting job to going full-time with it and another four to get to the point where I no longer worry about where the next client is.

Consulting can be lucrative, and it can be fun. But it’s not something you should go into without a solid understanding of what your finances are and how they’ll be affected. This article is at best an overview, one I hope will give you a healthy dose of caution. Start by looking at the issues raised here, but don’t go ahead based only on those issues. Invest some time and money in a tax accountant or financial planner. There are lots of good ones out there; find your favorite consultant and ask him or her for recommendations.

And best of luck.

**Making the Jump: Moving from Permanent to Contract Employment**

*by Dave Clark*

< dclark@mindsrc.com >

Today’s demand for talented UNIX system engineers has gone wild. The World Wide Web has placed even greater demands on this limited market. With so much demand, qualified professionals have many employment options. One popular option is contract work.

The use of an agency as a “middle man” in the job market is on the rise. Should you consider contracting? How much risk is involved? Although the contingency work force among high-tech companies is as high as 20% on some sites, contracting is not for everyone. Still, it is smart to pay attention to this growing trend.
Many people I interview enter the job market with no predisposition toward contracting or permanent placement. They look instead at the type of work and the various benefits associated with the particular opening. Often, the choice between contracting and permanent employment is secondary to other factors, such as commute, startdate, and how enjoyable the assignment will be.

Still, there are differences between contracting through an agency and consulting as an individual. I’ll focus on the bulk of today’s openings, which are W2 employment through an agency.

**Why Contract? Why Permanent?**

There are several reasons to pick up a contract. Sometimes the full-time jobs don’t come along. Contract hire process is often only a few days. Contractors always have a higher hourly wage and are paid for overtime (unlike salaried professionals). Good contract houses provide benefits such as health and dental insurance, 401(k) retirement plans, and athletic clubs. Looking for a job becomes routine if you have a good agency; you will become a more experienced interviewee in the process, however.

Permanent employment typically includes more generous benefits in the areas of training and vacation. Permanent employment packages may also allow for some big ticket items such as relocation. The medical benefits of a big corporation are generally better than those of an agency. The big difference in medical plans is probably going to be dental. And, of course, if things go as planned, you won’t have to look for jobs all the time.

**Current Trends**

Corporations of all sizes are turning to contract employment because it allows them flexibility in strategic planning: they can achieve rapid ramp-up and ramp-down times, they can shield themselves from tax and business liabilities, and, it offers a last resort when they can’t find the appropriate talent on a full-time, regular basis.

Employees are turning toward contracting because it gives them greater mobility: it provides extended leaves of absence between contracts (actually quite rare), it pays much better, and it shields them from corporate politics. There is a growing trend for consultants to take several contracts and then go permanent at a job they have worked at for a period of time. This is known as “contract-to-permanent conversion.”

**1099 vs. W2**

An employer can pay you as a business (1099) or as an employee (W2) for purposes of taxes. The guidelines for 1099 contracting are fairly rigid, and most contractors do not qualify for 1099 consulting. The 1099 implies a business-to-business situation with the contractor having multiple simultaneous clients, direct overhead, and an involvement in a project that involves assuming a risk of payment.

The two reasons that most corporations do not care to hire consultants on 1099 are because of direct tax liabilities and product liabilities. Bear in mind that 1099 consulting is not shielded by the National Labor Relations Board. When you go out bill collecting, you will do so as a business, not as an employee.

**W2 Employment**

W2 employment pertains to most individuals. It takes care of your taxes exactly like a full-time, regular job. In fact, it is just a short-term, regular, full-time job.

When you work for an agency, you’ll be paid for the hours on your time card, including overtime. Premium rates such as time and a half or double time generally don’t apply to individuals earning over $20 per hour. You may receive preferential treatment if you work many weekends or travel extensively while on a contract. If the client company wants to put a cap on your hours, it might provide comp time: time off from work later in the billing cycle.

**Moonlighting as an Inroad**

Many people start moonlighting as a way to take on different types of work and to earn extra money. However, full-time employee agreements may restrict this activity, though I’ve not seen any such restriction enforced. You’ll have the best luck finding your own part-time jobs through friends; most agencies don’t want to get into short-term, part-time contracts.

**Precautions: How Great the Risk?**

Because of the favorable market conditions, contracts are relatively easy to find. If you are moving from a full-time job to a contract, you can do so with a fair amount of comfort. First, you’ll have adequate time to shop the market. Second, if you work for a large company, you and your dependents will be covered by your existing health insurance via COBRA for up to 18 months. If you get a sour taste in your mouth with your first contract, you’ll still have time to pick up a second one or go back into permanent employment.

Should you choose to return to regular, full-time employment after a contracting stint, you’ll be able to use the leverage of your higher pay rate as a contractor in negotiating a favorable salary.
Pay Rates
The going rates for contractors vary wildly from one company to another. The low end for junior-level systems administrators is about $20 per hour. A talented senior-level sysadmin can earn between $55 and $75 per hour for a contract. Your mileage may differ. Consider factors like the location of the job (do you want to wear a suit?), the type of job (a cool R&D Web job will pay less than a hellish stint in an MIS group), and the duration.

Make Sure to Charge a Fair Rate
I commonly see situations where contractors are getting paid twice what they are worth because the agency is not capable of assessing the candidate's skills and because the candidate did a good sales job. The results are predictable. The contractor is unable to perform, bungles the job, tries to place the blame in a different area, and gets terminated nonetheless. Everyone loses, and the consultant’s reputation gets tarnished.

A Little Bit of CO2
Often people call in consultants to make problems go away. Accepting a contract that is over your head is asking for problems; this is not the place to learn. Avoid head nodding when people ask if you can fix complex problems that are endemic to the whole company. Remember, the work you do through an agency is hourly—not project based.

Giving Notice
Assuming you find a suitable contract, use common social courtesy to negotiate your departure. Don’t burn bridges, even with unsympathetic managers. Try your best to time your departure in some way that will not leave your organization hanging (like the day before a major product release). Two weeks notice is the industry standard for any job.

In California, either party can terminate employment at will, for any reason—this is something that permanent employees, as well as those who have turned to contracting, must bear in mind.

Starting Your First Contract
Your first contract will probably feel much like the first day on any job. There is really very little difference. Toward the end of the contract, you will feel more angst: time to look for another gig.

Jumping Back:
From Contract to Permanent
Many individuals take several contracts, and, because of a great offer or opportunity or because of other events in their life, choose to switch to permanent employment. The contract experience will stay with them, and the responsibility and challenges they took on will make them better employees.

How Long Will This Trend Last?
The market is going crazy now. Everyone who can find their way around the capslock key can get a job. There is no end in sight to the shortage of skilled UNIX systems engineers. Things change, however. I always advise people who are earning good money as a contractor to put some of it into retirement and savings. Plan for a rainy day as any good business might.

In the Silicon Valley, contractors are used when companies grow, when they shrink, and when they stabilize—pretty much all the time. I predict that the upcoming years will continue to see a mix of contracting and permanent employment opportunities. Even if the economy slows up, it will still be only a matter of time before the demand for contract employees bounces back in one form or another.

The present business climate presents good prospects for UNIX contractors, even if it is only a stepping-stone to another permanent job.

The Computer Consultant’s Workbook

Book review by Brent Chapman
<brent@greatcircle.com>

As someone who has made his living as a consultant for several years, I’m often asked questions about how I got started at consulting and how difficult it is. I’ve found consulting to be a very lucrative and rewarding way to make a living, but it’s not for everybody. If you desire a stable, predictable, low-risk work situation, then consulting isn’t for you; but if you don’t mind instability and aren’t too worried about where your next paycheck is coming from, then consulting can be a lot of fun.

I just picked up a great book called The Computer Consultant’s Workbook by Janet Ruhl. I wish I’d had this book when I was getting started as a consultant; it would have saved me
a lot of the time and trouble of learning the ropes. The book is well organized and full of great tips, useful advice, and excellent discussions of the issues involved in being a successful consultant. It was clearly written (and written clearly) by somebody who has obviously "been there and done that." As I read through it, I kept saying to myself, "yeah, that's so true, and that's true too, and I wish I'd known that before discovering it the hard way, and . . ."

The first chapter poses the question, "Are you ready for consulting?" It talks about what consultants do, why clients use consultants, myths and realities about consulting, and so forth. More importantly, it includes worksheets to help you determine whether you have the right personality, skills, and personal situation (family obligations, financial obligations, etc.) to make consulting feasible.

Further chapters discuss the business side of consulting (sole proprietorships vs. partnerships vs. corporations, business licenses, taxes, deductible expenses, insurance, etc.), setting rates, finding clients, working with brokers, winning jobs, and negotiating contracts. The last chapter, entitled "Be a Success on the Job," is full of tips and suggestions for actually making things work (i.e., getting yourself out of whatever you've gotten yourself into). Throughout the book, there are worksheets, checklists, fact sheets, sample letters, and sample contracts—all designed both to help you decide what to do and to help you do it.

If you've ever entertained idle thoughts about consulting in the computer industry, I strongly encourage you to get this book. It does a great job of presenting both the pros and cons of consulting and includes the essential information you need to succeed if you decide to make a go of it.

Fixing the Box

A new customer needed a Sun installed and didn't know how to do it. He found out that I knew how and asked if I'd do it for him. I installed it and there were no problems. Everyone smiled. They mailed me a small but welcome check.

This type of straight technical work with no problems, as we shall see, rarely happens by itself. Still, it's an important component of all projects. You must never lose your technical "edge" and must keep your technical skills current, or your consulting engagements will be overly frustrating, and you'll become technically obsolete.

Don't confuse "fixing the box" with consulting. "Fixing the box" is contracting. True consulting has many more dimensions, and requires the ability to bring a broader vision into play.

Fixing the Problem

A client needed Solaris installed on a loaner machine. I started the installation and ran into a problem with a dead disk drive. I contacted the vendor and convinced them that, even though the vendor didn't want to deal with disk replacement on a loaner machine, it was in the vendor's best interests to send my client a new disk. I explained this to the client and explained that the whole project would be delayed while we waited for the disk to arrive. The disk arrived and I completed the task. Everyone smiled. The client mailed me a modest check.

The difference in this project is that I was able to work with a cranky vendor to get what the client needed and able to set the client's expectations such that the client knew the project would be delayed but still had faith in my ability to get the job done.

This situation comes up often. All consultants should have at least this level of proficiency in "managing the client, the vendor, and the situation." Some contractors would have just told the client "The disk is dead. Call me when you get a new one." Such contractors will probably not hear back from the client.

Fixing the Real Problem

A client wanted her company to use PPP on a Sun for Internet access from all the company's desktop machines. She asked me to configure PPP for her. I realized that before they could have their full Internet access, they needed to reconfigure their internal routers more cleanly or no packets would arrive on the desktops. I explained this to the client, got the go-ahead to fix the router configuration, then configured PPP. Nobody was taking care of browser software, so I got approval to install licensed

Breadth of Vision—A Key to Successful Consulting

by Celeste Stokely
<celeste@stokely.com>

A successful move from full-time employment to consulting requires many skills beyond technical knowledge. One of the most important skills to develop in your client engagements is a "broad vision." Being able to see much further than the immediate technical problem results in lots of repeat business, referrals from all over the place, and an "in" to more interesting and better-paying work.

Below are some war stories from my years as a consultant that show how the scope of vision I had at the time affected the outcome.
Web browsers on all the desktops. Everyone cheered. They mailed me a fairly substantial check.

I saw that fixing the original technical problem (configuring PPP) was not going to solve the client’s overall problem of “Internet to the desktop.” I was able to explain this to the client and get the approval to work on the “behind-the-scenes” problem (the router configuration) and approval to purchase and install good Web browsers. Taking the “broader view” on the project gave the client more confidence in my ability to solve her total problem, and let me design a solution that made the client far happier in the long run.

There are many consulting engagements that require this level of “tactical vision,” even if neither you nor the client realizes it at the outset. This level of vision and proficiency will get you lots of work at middle-tier rates. It will also bring in some referral work, though not as much as you may want.

This is where many new consultants’ scope of vision stops: at the technical level just beyond their noses. But it’s not where the opportunity stops.

Now let’s move on to where the work is more involved and the rates are more rewarding. But first, a word of caution.

“Solving the whole problem” is valuable if and only if that’s what the client really wants. Otherwise, it’s cowboyism that can lead to disastrous engagements. Never, ever, change the scope of an engagement without the full understanding and approval of your client.

Fixing the Department

A client called me in to solve a problem in which the UNIX users were hogging the Novell printers—she wanted to deny UNIX users access to the printers. I realized that this simple technical problem was masking a larger managerial problem—the computing resources were so scarce and laid out so poorly that everyone was fighting over them.

I worked with her MIS group to define a new architecture for the computing environment, giving different workgroups the types of fileservers and printers they needed to be effective. I helped them implement the solutions and helped train the staff in the maintenance of the new systems.

The company began the long road to a better computing environment, the MIS staff was happier, and the users loved the improved access to the printers. This client has now been with me a long time because I work with them to make everything run more smoothly for the company as a whole. A dinner was held to mark the successful completion of a long and involved project. They mailed me a very nice check.

Mastering this level of vision is the first step on the path to making serious money in fascinating realms of consulting. This stage separates the contractors from the true consultants and is the beginning of what most clients hunger for in their consultants—a broader strategic view of the customer’s business and the ability to develop ways that the customer’s business can be conducted more easily, saving the customer money in the long run.

If you have mastered this range of vision and chosen your market well, the phone will begin to ring frequently with referral and repeat business.

Fixing the Business

It was the dead of winter. Demo time was coming for an ISP, and it was for all the marbles. The president of a fledgling network service provider company in the Midwest called to say his engineers were fussing around with the production servers and now the servers crashed every hour. They were rushing to complete the working system to show to their investors, in the hopes of receiving another round of financing. A quick on-the-phone debug session with their engineers didn’t show me the problem. He asked if I would please fly there, stay two weeks, and get them running. I didn’t want to go live in snow for two weeks, so I quoted him an obscenely high daily rate. To my surprise, he agreed. So the next morning I hopped on a plane. As soon as I arrived, I found that the servers’ swap partitions overlapped the root partitions (they had not been entirely truthful over the phone when I asked about this), and they had trashed many configuration files. I fixed the errors, stabilized the servers, and the company was back on the air. But was it?

It didn’t take long for me to realize that the engineers were developing new code directly on the production servers whenever they chose, installing new libraries with no coordination, not working to any plan or guiding principles, barely able to code in C in some cases, highly frustrated, and fighting over too few terminals to use.

I explained this all to the president, who was still so grateful over the servers being up that he listened to me closely. I told him that, although I’m no expert in release engineering, bug tracking, or configuration management, I had more experience in it than I saw there; and I was willing to help in any way to make engineering run more smoothly.

In two weeks, we set up the development machines, taught everyone how to use RCS, instituted daily bug-tracking meetings, bought books on C programming that became required reading, and put someone in charge of...
testing and releasing new code onto the production servers. I also helped the president see that he was trying to build a world-class engineering organization out of talent he could find in a small Midwest farming community, and that’s difficult. I helped him understand that if the company were somewhere that was more “engineering resource rich,” like Silicon Valley, he’d have an easier time attracting professional developers and system administrators.

I stayed the two weeks. They shipped their product nearly on schedule, got their venture funding, went public, moved to Silicon Valley, tripled their staff, and landed several multimillion-dollar accounts. This was some of the hardest work I’ve ever done, because it was a constant stretch of my capabilities. I burned up the phone wires talking with experts I knew in each field, staying a few chapters ahead of the client. I documented my findings and the work we accomplished during that time, along with my recommendations for future action. It helped me grow enormously. They mailed me a truly gratifying check.

Without someone like me being there at that tender time, that company might not have gotten funded. I have received many excellent referrals from them in the years since.

It takes years of trial and error to develop this farther-reaching scope of vision, but it results in more satisfying work, work that is easier to obtain and pays in the upper tiers of the profession.

Referrals keep you from having to do what is easily the worst part of consulting—making cold calls to try to find work. When you behave as the professional’s professional, you become something of a client magnet. They will find you.

Plus, when you have clients who value a professional’s vision, they’re often good professionals themselves and far more pleasant to work with.

**How I Got Here**

I suppose I got the desire to own my own business from my mother. She has been self-employed my whole life, owning a series of beauty shops and, later, leasing stations in other salons. She’s always worked hard but enjoyed the perks of being her own boss. The freedom she had as a result of her self-employment wasn’t lost on me.

After time in the Navy, jobs as a technician for a consulting firm, an SE for a super-mini-computer vendor, an OS developer for a series of Silicon Valley companies, and a system administrator, I began to get the itch. I had a marketable set of skills, some money in the bank, and a network of business contacts that (I hoped) would lead to a steady stream of consulting clients. So, in time-honored tradition, I returned from my sabbatical in June of 1995 and formed Mellis and Associates. (At that time my only “associate” was my golden retriever, Cinnamon.)

**What I Do**

When I started out, I thought most of my work would be in Internet connectivity and network security. It has turned out to be somewhat broader than that. I’ve done a little bit of everything, from ordering counter tops for a printer room to configuring routers in Switzerland. I’ve chosen jobs because they were interesting or because they would give me an opportunity to learn about a new technology.

I’m an implementor, not an inventor. I once described myself as a network carpenter, as opposed to a network architect. Master carpenters have an eye for structures that work. They’ve built enough walls to know the working strength of a two-by-four, and they know when to call in a structural engineer for the tricky stuff. Similarly, I’ve seen good solutions to engineering problems and bad ones. I study successful designs so that I will be able to reproduce them for my clients, and I’m not too proud to call on an expert if I need bleeding-edge knowledge.

**Lessons Learned**

I’ve had several jobs where I was set up to fail. A consultant is an easy person to stuff into an untenable position—implementing an unpopular policy, for example, or wiring a network on a pie-in-the-sky schedule. I’m learning to avoid these situations, and I try to take only jobs that I know I can complete successfully. Why stack the deck against myself by taking jobs that are doomed from the beginning when there’s plenty of risk in even the easy tasks? My reputation is my most valuable asset, and taking jobs that I know I can’t do is professional suicide.

My consulting practice has reinforced my pragmatism. Clients pay for results more than elegance, and sometimes I just
have to use an esthetically repulsive technique to get the job done, no matter how much it hurts.

My customers tend to be seasoned real-world engineers and engineering managers. I've found them to be understanding when I have bad news to deliver, like "it's taking longer than I expected" or "I made a mistake." I do my best to be frank with my clients, and they do their best to approve my invoices.

Something I didn’t expect when I went out consulting was loneliness. In my "real" job, I was a member of a close-knit team, and I missed that team environment when I left it behind. Since then, I’ve cultivated friendships within my new peer group—other consultants. Being the gregarious fellow that I am, I derive great pleasure from calling up another consultant to whine about the bizarre sendmail configuration I’m working on or just to hear a friendly voice.

I still worry that the day will come when I don’t have anywhere to go the next morning. My consultant friends who have been in business for a while tell me about the slow times and advise me to save for them. When I combine “slow times” with my innate self-doubt and my need for affirmation, I end up overcommitting, and I’m miserable because of it. This is my biggest problem today. However, I’ve been busy with longer term projects, and I’m feeling confident enough to say “no” more often and spend more time with my family (A Good Thing). And, after all, if things get too bad, I can always get a “real” job.

My Own Boss

by Shawn Instenes
<shawni@celene.rain.com>

I began consulting a few years ago, on a part-time basis, working around my “real” job. I gave the idea of full-time consulting serious consideration when, one fateful month, I realized that I’d almost earned more consulting “on the side” than I’d earned at my regular job. So now I’m doing it for real.

Consulting life has an amazing set of upsides and downsides—most of them obvious, but some are subtle. Here are some things I’ve observed about consulting:

1. I like being my own boss. I’m no longer subject to the whim of upper management, at least in my own company.

2. I am paid better for the same work I used to do as an employee. I also have more expenses (such as health insurance, lawyer fees, a bookkeeper), so this isn’t quite as wonderful as it sounds.

3. My income arrives in an extremely irregular way. One month the money may be pouring in (remember those taxes), and then I might go two or three before anyone else pays me. Some companies think “net 30" is more like “net 90.” Income flow like this requires an adjustment much like switching from weekly paydays to monthly paydays, and a fair amount of careful budgeting.

4. My hours are very flexible. I’m not at all a morning person. Often, I can arrange work times around this preference.

5. I work longer hours when I do work. Rather than a steady flow of 40-hour workweeks, my workload varies from 10 to 60 billable hours per week. Of course, there are hours I work on contracts, meetings, and proposals that aren’t billable.

6. There is more paperwork: nondisclosure agreements, performance contracts, work proposals, making estimated tax payments, and so on.

7. The bane of my employee existence, the dreaded pager, is back with a vengeance. It beeps rather more frequently. However, I am paid extra each time it goes off. It’s harder to schedule vacations, conferences, or evenings out with friends.

8. I have a 30-foot commute for those jobs on which I can work at home over encrypted data links. This has all the advantages and disadvantages of telecommuting—I can be distracted from work by other things (my cats, usually), but the work environment is more pleasant than most offices. I can crank the stereo.

9. Clients have high expectations of consultants. A consultant is brought into a job because the job is known or expected to be beyond the capabilities of the client (either technically or, less often, because of office politics). If I don’t feel that I can walk in and take care of the job quickly, I’ll recommend someone else I think can do it.

10. Part of being a consultant (for me) is to know where the strengths of fellow consultants lie. I am a member of an informal organization of consultants that provides two services: we refer clients to one another, and we cover for each other when a member wishes to go to a conference or go on vacation. Our customers are much happier to have full-time support, and we tend to get a lot of repeat business.

I’ve listed a mixed bag of perks, peeves, and opinion. The fact is, consulting is a risky way of making a living; and I’d never have started without lots of encouragement from friends, business partners, and clients. Now? I enjoy the challenge.
Interview with Andrew Hume

Rob Kolstad interviewed Andrew Hume, USENIX's incoming president, via email.

Rob: Welcome aboard as incoming USENIX president. Do you have specific plans and ideas you'll be pushing to implement as soon as you take office?

Andrew: Thanks. Right now, USENIX is in good shape: it has a respectable (and increasing) number of members, very healthy finances, a good slate of conferences, and an excellent staff. Yet, as the computer world changes, as the number and types of users change, as the technical challenges in serving those users change, so does the need for USENIX to refine and adjust its technical focus; and in doing so, USENIX might want to alter its membership makeup. To be specific:

- USENIX needs to reassess how big it wants to be and what its primary goal ought to be (as an organization).
- USENIX needs to decide if it wants to increase the diversity of its membership. As an example, we are actively pursuing users of PC-based UNIX systems.
- USENIX needs to implement fully, and perhaps expand, its new program of proactive student outreach.

Rob: USENIX's financial health is the best it's been in decades. Do you have ideas on how best to use moneys to advance USENIX's goals?

Andrew: Both the previous and new boards have had good ideas on how to invest USENIX's profits (substantially due to the competence of the USENIX staff and to Rick Adams's generosity and stewardship as treasurer). I have been and will continue to be an advocate for funding student- and standards-oriented initiatives. This includes:

- student scholarships
- increased fund for student stipends for attending conferences
- creation of a fund for student grants for Computer Science projects
- extend our standards coverage to include the IETF (which moderates most Internet related standards)

Rob: Are there any moves afoot to expand the general conference offerings back to two per year?

Andrew: I have none. While I regret the upcoming 18-month gap between main conferences, I am very pleased at the greatly improved quality of the technical papers at the general conference. I think this is largely due to the fact we have only one general conference per year.

Rob: Do you see USENIX moving its focus to that of the Internet and Web?

Andrew: It already is! Of course, it's more complicated than that. As Mike O'Dell once said, USENIX's job is moving information from where it is to where it isn't. We currently do this in a number of ways, including conferences, workshops, ;login:, Computing Systems, SAGE pamphlets, and reprintings of documentation. The key issues are focus, content, and delivery.
USENIX's focus is on three overlapping groups: CS researchers, OS and application developers, and system administrators. I have been a staunch advocate of extending our focus to include the growing PC-based UNIX community, which includes the various BSD-based systems and Linux. One direct consequence of this is the Linux-oriented activities at the upcoming Anaheim conference.

Contentwise, USENIX has been doing a pretty good job of giving its members the information they need and anticipating the information they will need. Given the ideas generated at the last board meeting, I expect USENIX will continue to do well.

USENIX has the following main vehicles for delivering information: the general conference and LISA, specialized workshops, ;login:, SAGE booklets such as "Job descriptions," and the journal. Our conferences are well received, the workshops are popular, and the SAGE publications are valuable resources for the SAGE members. USENIX is working on various proposals to improve ;login: (which I think is doing a fine job). There is significant concern over the journal and the small number of submissions; the board is monitoring the situation closely.

Returning to the original question, USENIX has had many talks, papers, and tutorials on the Internet and Web (Java is a recent example). And we'll continue to do so as long as the membership needs and wants them. But there are a lot of other trees in the forest.

Rob: Some have observed that Microsoft's incredible market share of PC operating systems might have a deleterious effect on UNIX. Do you see the end of the road for UNIX any time soon?

Andrew: PCs are great. The speed and price/performance of modern Pentium-based PCs really drive home the advantages of using commodity hardware. And, best of all, you can get real software to run on them. While I have been using PCs for a few years, over 90% of my use has been non-Microsoft (mainly Plan 9— I play Doom on my wife's Macintosh).

My personal assessment of PC software is that nearly all of it is at best mediocre. And where are all the people who have complained about how hard UNIX is to use? Surely they would have something to say about the vortex of despair waiting for anyone wanting to reconfigure their PC. I have experienced my fair share of bugs and disasters during my prime UNIX years of 1975-1985, but none of it was as futile and frustrating as a recent attempt to change a Windows 95 PC running Exceed to use a three-button mouse. Plug and play and pray and play . . . . (The solution? Keep a supply of small mammals to offer as sacrifices, and maintain a very close relationship with a PC guru.) After living the Windows 95 experience to the full, I anticipate following my col-

leagues who run real software (Plan 9 or Linux or some other UNIX-like system).

UNIX, and its derivatives, has long been the dominant environment for OS and application research because of its ubiquity, familiarity, power, and accessibility. In general, I think this will continue to be true for the foreseeable future. Microsoft's dominance of the PC software world has little bearing on this; mostly, the problems stem from third party suppliers who won't tell you how to program their devices but supply a Windows driver instead. On the other hand, I am not wedded to UNIX. I can, and will use, almost any environment that helps me do my job well, and should that mean using some Microsoft operating system, then so be it.

Musings
by Rik Farrow
<rik@crow.spirit.com>

I just got back from Chicago, after being tricked into going to Spring Comdex. I had, innocently enough, agreed to chair a panel entitled grandiosely "The Future of Internet Security" and then to teach a two-day Internet security class for UniForum. What I didn't realize was that this was in conjunction with Comdex, which wasn't in Las Vegas this year.

I didn't see all of Comdex, just a small corner of McCormick Place. I don't get off on being in crowds. I did get to see more of Chicago, which turns out to be a nice city: lots of Thai restaurants, not to mention loads of Italian restaurants. The Art Institute has a world-class art museum, and there's a section of stuffy private galleries, some of which have great art in them.

I mention Chicago because of my other reason for being there. LISA 10 will be in Chicago in October, and the Program Committee met on June 8 to pick papers and invited talks. As co-coordinator of invited talks, I got to listen in on the process of choosing papers and would like to share a little of it with you.

Committee Time

Ever been in a meeting that lasts forever and accomplishes nothing? Well, the Program Committee moved fast and furiously, accomplishing its goals in a single day. Maybe being successful UNIX system administrators has taught many of us how important it is not to waste time.

Before we reached Chicago, members of the committee had each been assigned about 40 abstracts to read, with five reviewers assigned to each of the 78 submissions. The reviewers rate each paper for relevance, presentation, and quality (which includes accuracy). The reviewers also rate
themselves according to their skill level in each paper's topic. These scores are then sent back to the Program Committee chair, who uses a Perl script (there's an Awk version too) to compute averages, and also variances (a measure of the variety in the reviewers' responses).

The scoring helps the process a lot. Papers that scored low, and had low variance (which meant everyone agreed), could be dealt with quickly. Even so, each paper was individually reviewed, given a second chance as it were. The most common comments were "no references" or "author appears unaware of prior work." "Goal of paper not clearly defined" and "solution of problem not included" were also the ends of several papers. By lunch, the committee had worked through all the papers in the lower half, stopping when they reached the "maybes."

After lunch, the process began again at the top. All papers, with one exception, in the top 20% were accepted. The only exception was Peter Van Epp, who got an "Invited Talk" slot to discuss his experience with ATM as compared to other high speed networking technologies. Choosing the next 12 papers took longer, because abstracts were closer together in quality. When there was a pile of "maybes" and only three slots left, a new approach was taken.

**Piles**

A co-chair had been quietly sorting the selected abstracts by topic. Another committee member at this point became inspired to do the same thing—very high tech. Groups formed around the piles of papers on the floor toward creating sessions with related papers, while calling out changes to different piles.

If a pile was short, the "maybes" were checked for a good match that was also a good abstract. Eventually, all the slots were filled and each "pile" assigned to a day and time. Now it was my turn to help select invited talks that wouldn't draw the same audience as the session topics.

My wife, who watched some of the proceedings, commented on how civilized and efficient the whole process was. I was grateful, having participated in many meetings that achieved far less in the way of results. Again, I have found that, in general, people in the USENIX community are good to work with. A nice, warm, fuzzy feeling.

**Audits**

Changing gears here. I got to meet Dan Farmer again recently. Dan and Wietse Venema gave a one-day workshop on April 30 focused on auditing UNIX systems, networks, and firewalls. Sun Microsystems sponsored the talk—all you had to do was surface mail your request to attend and show up in Santa Clara, CA. The workshop was fun and interesting. Dan and Wietse have plans to write a book about auditing UNIX-type systems and networks and wanted to test some of their material. Wietse included an audit of Bill Cheswick's new firewall (well, perhaps I should say Lucent's). The "new" firewall is a packet filter, and Cheswick allegedly complained about the GUI interface. Wietse also included an audit of Dan's home network, presented as a complete audit report. I personally wondered if it was a good idea to publish both Dan's address and the fact that his sliding glass door is often left open.

When I talked to Dan after the workshop, he said to me, "Well, you're a writer, aren't you?" I was somewhat taken aback, because writing is only a small part of what I do. You might say that learning that I could write somewhat coherently was a "happy accident," like the splashes of paint on canvas that made Jackson Pollock famous, rich, and somewhat suicidal.

**Writing**

When I got out of college, I hated writing. I had managed to graduate having written only one real paper and had no desire to write. The first time I attempted any technical writing, I threw it away immediately after I read it. I really thought I could do better than the stuff the engineer had written, but I was wrong.

Later, as a hungry, self-employed programmer, I accepted the task of revising a manual for a disk controller. The manual was already on disk, and all I had to do was convert it to represent an I/O-mapped rather than a memory-mapped interface. Piece of cake, and easy money. This lead to ghost writing, then to book contracts, and fame without fortune.

I started writing reviews as another "happy accident." The programming shop where I was contracting had just gotten a new Silicon Graphics workstation, and I called up a magazine offering to review it. Workstation reviews became a paying hobby, and for years I had the latest hardware sitting in my home office. I still suffer for this, because I was always a nomad and never had a stable environment.

I mention this now because I'd like to get into the reviews business again. As I got more serious about reviewing, I began to realize that there was a lot you couldn't do at home. Benchmarking required a stable environment and more time than I could afford. I looked into professional labs, thinking they would do a better job. It turns out that most magazines' reviewing labs are barely adequate—the lab personnel must come up to speed on a new technology, install and test it, then move on to the next topic. Jacks-of-all-trades and masters of none. It's amazing how good they can make their reviews sound.
Besides, I wanted to review the really interesting stuff: network managers, supercomputers, tape silos, network switches—things you should really test on-site, not in a phony lab environment. I came up with the idea of finding people who had recently acquired the new hardware and/or software and working with them to produce a review that was more of a case study. Why did they choose this product? How easy was it to install and get working productively? How was your experience with the vendor’s support? Did the performance meet your expectations? Anything really weird happen during testing?

I’m still interested in doing this type of review. I really don’t want to install an FDDI testbed, or a really hot server (it’s already too hot here in Arid-zona), in my home office. If you are about to install (or have recently installed) a product you think might fit in this category, send me some email. Perhaps together we can write and publish a “case study” review in ;login:.

---

**Linkage and Symbolism**

_by Quentin Fennessy_  
quentin@benson.amd.com

What is a link in a UNIX filesystem? For readers who are new to UNIX, this may be a confusing question because there are two (or more) answers.

At the simplest level, a link is an entry in a directory that refers to a file. A file may have one or more links pointing to it. There are specific differences and limitations between the two main types of links.

**Hard Links**

A hard link is an elegant feature of the original UNIX filesystem. The other kind of a link is called a symbolic link, symlink, or soft link and was added in 4.2BSD UNIX.

When you refer to a file in UNIX, you will typically use a file name. A fully qualified name starts with /; otherwise, name resolution assumes a current working directory as a starting point to resolve the name. A program that opens a file such as `uptime` can either use an absolute path such as `/bin/uptime` or assume a working directory of `/bin`.

In fact, this filename is actually a link to an inode (inodes describe the rest of a file's statistics and tell how to find the file's blocks on disk). Generally, a directory entry includes a filename and inode number.

Because the directory entry contains an inode's number, any given inode might have several different names listing its inode number. So the file named `/bin/uptime` may have other names that refer to the same data on disk; `/bin/w` is such a link on my system.

The `ls(1)` command is very useful for finding much of the information stored in an inode. Here is an `ls -l` listing of two files:

```
$ cd /bin
$ ls -li uptime
15288 -r-sr-xr-x 2 root bin 11192
Oct 25 1995 uptime

$ ls -li w
15288 -r-sr-xr-x 2 root bin 11192
Oct 25 1995 w
```

The first number on each line of `ls(1)` output is the inode number. Because the two numbers are identical, you can see that `uptime` and `w` refer to the same inode. The third number is the number of hard links that refer to the same inode (two in this case).

To create a third link to this program, do this:

```
# ln /bin/uptime /bin/u
```

Then observe the results:

```
$ ls -li uptime u w
15288 -r-sr-xr-x 3 root bin 11192
Oct 25 1995 /bin/u

15288 -r-sr-xr-x 3 root bin 11192
Oct 25 1995 /bin/uptime

15288 -r-sr-xr-x 3 root bin 11192
Oct 25 1995 /bin/w
```

The number of links has now increased to three. Interestingly, the date associated with `/bin/u` is the same as the other links (because the new name index is the same old inode). But the change times for the three different names (one inode) are new:

```
$ ls -li c uptime u w
15288 -r-sr-xr-x 3 root bin 11192
Jun 5 10:44 u

15288 -r-sr-xr-x 3 root bin 11192
Jun 5 10:44 uptime

15288 -r-sr-xr-x 3 root bin 11192
Jun 5 10:44 w
```

The `-c` option to `ls(1)` displays the time of last modification to the inode and shows you exactly when I wrote this paragraph. The inode was modified because an additional link was added, increasing the link count.

To identify a file uniquely on a UNIX system you need more than the inode number—inode numbers are unique only per file system. Uniqueness is only assured when you also specify a device on which the inode exists. This exposes the major limitation of this kind of link—if a hard link refers only to an inode number and does not mention the device, then
FEATURES

hard links will work only within a filesystem. You cannot use hard links between filesystems.

Symbolic Links

A new type of link called a symbolic link was added several years ago to thwart this limitation. A symbolic link behaves as a pointer to a file but may point to files on other filesystems. Rather than pointing to an inode as hard links do, a symbolic link refers to a string that is typically interpreted as a UNIX path. Here is an example of how to create and view a symbolic link:

```bash
$ ls -l Mail
    drwxr-sr-x 2 quentin cts 512 Jun 5 11:06 Mail

$ ln -s Mail mail.dir
$ ls -li
    921792 drwxr-xr-x 2 quentin cts 512 Jun 5 11:06 Mail
    737441 lrwxrwxrwx 1 quentin cts 4 Jun 5 11:05 mail.dir -> Mail

$ ls mail.dir
    sysfolder # looks like Mail!
```

From the example above you can see that a symbolic link called `mail.dir` was created. This points to `Mail` and, when it is referred to by `ls(1)`, it acts like the directory `Mail`. From the `ls -li` output, you can see that `Mail` and `mail.dir` have distinct inode numbers.

There are two implementation styles of symbolic links—the differences are slight and exist only in the implementation. Most applications and users will never notice the difference. Original symbolic links stored the target of the link in data blocks on disk-so a symbolic link required both an inode entry and a data block. Considering most symbolic links probably point to short paths, this is a waste of resources—a full disk block as well as the effort necessary to read the block in from disk.

Some UNIX systems will store the symbolic link path in the inode itself rather than in a data block if the path is short enough. The limit on Solaris 2.5 is 56 bytes, which is probably long enough so most symlinks do not require a data block.

Symbolic links are very flexible. The `ln -s` command creates a directory entry that is a string (that names the actual file to be accessed). If it is a reasonable pathname, then most UNIX programs will typically follow the link and treat it as the file or directory actual in the string. Symbolic links can be relative paths, so you can link to `.././lib`, thus preserving the semantics of the link even when a symbolic link is resolved using a networked filesystem.

Applications and Links

Questions about how applications deal with links often arise. Hard links are handled in a straightforward manner—all UNIX files are hard links, and so this is the definition of normal behavior. Symbolic links are trickier:

Q: If you `rm(1)` a symlink is the target removed?
A: No—`rm(1)` will only remove the link.

Q: Does `tar(1)` follow symlinks when creating an archive? `cpio(1)`? `find(1)`?
A: In general, these programs do not follow symbolic links. This is handy in case you have a link to `/usr/src/x11R6` in your home directory and you use `tar(1)` or `cpio(1)` to archive it. That archive could become very large.

Specifically, there are usually options to these commands to make them follow symlinks. See your man pages for details for your systems.

<table>
<thead>
<tr>
<th>Command</th>
<th>Solaris 2.5</th>
<th>HP-UX 10.10</th>
<th>Linux 1.99/GNU</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tar</code></td>
<td>-h</td>
<td>-h</td>
<td>-h,--dereference</td>
</tr>
<tr>
<td><code>cpio</code></td>
<td>-L</td>
<td>-h</td>
<td>-L,--dereference</td>
</tr>
<tr>
<td><code>find</code></td>
<td>-follow</td>
<td>-follow</td>
<td>-follow</td>
</tr>
</tbody>
</table>

Q: Can you `chown(1)` a symlink? Do you want to?
A: On Solaris 2.5, `chown -h` will change the ownership of a symlink rather than the default of chowning the target of the link. Don’t bother. Permissions and ownership do not matter on symlinks.

Q: What does `file(1)` report about symbolic links?
A: On Solaris 2.5, `file(1)` will follow the links and reports on the target of the link. But `file -h` will not follow links. There are exceptions—if the symlink is not a real file, then `file(1)` will report it as a symlink.

Q: Can you `stat(2)` a symbolic link?
A: A `stat(2)` of a symbolic link returns inode information on the target of the link. `lstat(2)` will tell you about the link itself.

Q: What tools are specifically useful for hard and symbolic links?
A: HARD LINKS

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ln(1)</code></td>
<td>Creates a link (shell command)</td>
</tr>
<tr>
<td><code>link(2)</code></td>
<td>Creates a hard link</td>
</tr>
<tr>
<td><code>stat(2)</code></td>
<td>Retrieves inode data of a hard link</td>
</tr>
</tbody>
</table>
SYMBOLIC / SOFT LINKS

ln(1) -s  Creates a symbolic link (shell command)
symlink(2)  Creates a symbolic link
readlink(2)  Reads the value of a symbolic link
lchown(2)  Changes owner and group of a symbolic link
lstat(2)  Retrieves inode data of a symbolic link

BOTH

unlink(2)  Removes a hard or symbolic link

Q: What system limitations exist for links?
A: Users may not create hard links to directories—only files. Otherwise filesystem loops may cause terrible confusion. Users may create symbolic links to directories, but this will be detected by MAXSYMLINKS (see next paragraph).

On POSIX.1 systems, symbolic links must be less than MAXSYMLINKS long in order to prevent loops, should be defined in params.h, typically equal to 20. The error ELOOP refers to symlinks that surpass this limit.

There is no substitute for reading the system documentation and experimenting with system utilities. If you have a question about system behavior, the combination of the two will probably help you. If your system is reasonably configured, you can probably use apropos(1) or man -k to research your system behavior. The system include files are also valuable.

The Cybermeds’ Invasion of Earth: Patients and Physicians Online

by Emily W. Salus
<esalus@bgsuvax.bgsu.edu>

Thursday through Saturday, May 30 through June 1, Harvard Medical School Continuing Education and Beth Israel Hospital presented a course called “The Computer as a Patient’s Assistant.” As the token “patient” among the presenters pointed out, the course should have been called “The Computer as a Doctor’s Assistant.” Most of the presentations focused on how computers could be used to elicit information from or transmit information to patients and their families without the trained medical staff having to spend time. The goal seemed to be creating ways for doctors to avoid interaction with patients. About 110 doctors, nurses, health administrators, health educators, and public health specialists attended.

Last year there was apparently much complaint that patients were not included, so this year the token patient (who spoke last year as well) was given 30 minutes instead of 10. In addition, the founder of Oncolink, the cancer information Web page (http://cancer.med.upenn.edu) and another medical Web page devoted to diabetes (http://www.lilly.com/diabetes) spoke about the information available to patients with serious or chronic diseases.

Thursday, “The Patient as Clinician,” consisted of an introductory talk by Warner V. Slack, MD, the director of the course, a keynote speaker (Sidney M. Wolfe, MD, director of the Public Citizen’s Health Research Group), and six presentations. On Friday, “The Patient and the Clinician,” there were six more presentations, and on Saturday, “Patient-Computer Dialogue,” there were four. Each day included “workshops” after the presentations. Two or three speakers would gather in a room with attendees interested in their subjects and answer questions. After this, all the day’s speakers would gather once again to respond to more questions.

Though many of the presentations addressed data collection or information distribution through terminals in clinics and/or homes, others addressed network use in medicine. Speakers from Beth Israel Hospital seemed particularly proud that they had both electronic mail and computer-based patient records. There was little discussion of security issues.

Sidney Wolfe’s keynote address, “The Patient as Consumer,” was less than fascinating, but the issues he raised are fairly important. The nonprofit organization he directs publishes a book that provides consumers with information regarding rights to medical records and the state laws surrounding them. Although the organization agrees that there should be federal law regulating record confidentiality, they believe the current Medical Records Confidentiality Act of 1995 provides less than adequate protection. The bill overall encourages the establishment of medical records data banks (read: computer files) that would allow, so the Public Citizen’s Health Research Group believes, many to gain access to medical records without the patient’s consent. Access to this information by current or future employers, insurance companies, “health oversight agencies,” public health authorities, health care researchers, and government law enforcement officials could lead to unfair hiring practices, inability to acquire medical coverage, or other persecuting practices. Wolfe suggests in a handout that computerized medical records imply a far greater need for patient confidentiality and careful use of patient identifiers in computerized medical records. Other than the handout, little regarding security issues was discussed.

One physician informed me that computer security was merely a subset of confidentiality, but in an informal discussion between sessions, a health educator and administrator...
told me that information about medical legal and privacy issues could be found at http://www.epic.org.

Most disturbing was the denial by almost all the attendees that security was an issue. When asked about email interception and forgery and the ability to break into records through Internet-connected machines, the many MDs and PhDs responded that they “didn’t think it was possible,” were sure that “encryption” was satisfactory, and that “it was possible to break into the machines, but who would?” The comment about encryption abilities immediately followed a comment from a doctor who said that he didn’t like email because the email address had to be exactly right (no typos). There seems to be a difference of opinion about the knowledge about computers that doctors have, even among the doctors themselves. Still, no one seemed to notice the discrepancy between one physician complaining about the accuracy needed by computers and another talking about security using encryption.

One speaker from Beth Israel, Daniel Z. Sands, MD, noted in his presentation that he exchanged email with patients about both sensitive and nonsensitive issues and that, if he referred a patient to a specialist, he sent his notes and records to the specialist via email. I didn’t inquire whether Beth Israel had asked the patients’ permission to send their records in this way. After being told in the presentation that the issues around confidentiality and the computer were (1) Did the doctor receive it? (2) Did anyone else read it? (3) Did the response get sent? (4) Is email between a doctor and patient part of the medical record? (5) Is email legally discoverable? Without any further engagement with these issues, I was afraid to ask. The only “inappropriate” uses of email were said to be medical emergencies, time-sensitive materials, and negotiation issues where much discussion was needed between practitioner and patient.

I was told that patients knew the risks of being patients and “chose” to make themselves vulnerable to that, just as doctors knew the risks of being doctors. I was also told that patients knew the risks of security issues on the computer and email. Given that the doctors themselves told me security wasn’t a problem, I rather doubt that patients are aware of the problems.

The only two people who acknowledged my concern with security were a physician from Jerusalem and the token patient, who won my vote as most aware computer user. The token patient was a woman who started and runs BRAINTMR, a listserv for those who have or had and those assisting individuals with brain tumors. A huge proportion of this list’s members (almost a quarter of the subscribers) are medical professionals.

Many doctors were happy to proclaim that they knew about “smileys” and “emoticons” as though they had discovered something new. Others proclaimed about the efficiency and ease of email, while some attendees questioned email as just another thing to take up their limited time. The differences in familiarity with the technology were astounding.

Primarily, the Web was seen as a place for patients to get information (http://www.medinfo.org, for example). Although PaperChase was mentioned as a source for articles and information for both doctors and patients, there seemed to be a segregation between what was appropriate for each population. In his presentation, one physician informed the audience that computers could be used to “teach patients and their families what we want them to know.” No one asked what patients and their families wanted to know or what they wanted to do with the knowledge. Patient knowledge seemed to be a tool used by doctors so home health care nurses didn’t need to be sent to care for patients and money could be saved.

Much of the conference was about money. HMOs and managed care were at the top of the list. Most of the physicians were affiliated with at least one HMO. Only one speaker, an RN/PhD, pointed out that some people’s homes didn’t have doorknobs and showed a slide of a park bench. Aside from that, no one addressed the homeless, illiterate, or those without health care at all. When I asked what would be done with patients who couldn’t read enough to use a computer, I was told that (1) a patient advocate could assist and (2) voice recognition software could handle that. Somehow, I just don’t think that poor, inner city, public hospitals with few computers in the first place are going to be able to afford voice recognition software (as well as expensive computers). The speaker himself informed the audience that when he went to Boston City Hospital in the late 1980s, the hospital had only one computer, and it was used for lab analysis. So much for the excited discussion about making these computer diagnostic programs and home health care computers available to everyone equally.

Some of the diagnostic programs looked fascinating, but were not in the exhibition area. As a result, it was hard to get a real sense of the programs’ capabilities. Richard G. Rockefeller, MD, had a really interesting program for both diagnostics and management of specific problems. The program took symptoms, rather than diseases or illnesses, as the starting point. The patient could start with “chest pain” and answer a series of questions that would ask about other symptoms. Ultimately, the program would produce a list of possible diagnoses and report how many of the questions the patient had answered that fit the criteria for the condition (i.e., 8 out of 21 for anxiety, stress, and depression). At that point, the patient and doctor could review some of the questions again or could pursue physician-guided analysis and physical examination to arrive at a diagnosis.
Rockefeller admitted one of the program’s largest weaknesses: so far there are only 60 “problems” included, they are not the most common problems, and about 8 of them are psychological. Why? Because the individuals who work on the project create questions about the illnesses that identify their particular research interests. Because a psychiatrist got interested in the program and did the work for those 8 problems. Still, there’s a lot of potential. There is a basic health status and a mental health status question series.

Rockefeller noted that doctors used to have skills in patient examination, empathy, and humor, as well as other areas, but that all that space in their brains was now occupied by biomedical knowledge, the only thing medical students are examined on. He proposed that diagnostic computer programs could be used to alleviate the burden of memorization that physicians experience and enable them to have more training in other, patient-centered skills while the computer stored biomedical data. Sounds good to me, but other speakers noted that computers would mean less face-time for patients (and seemed to think this was a good thing).

Granted, the conference was at least theoretically about “the patient,” but there was no discussion about the Internet as a tool for doctors to use to share or transmit information, except where specific patient information was concerned. There was no discussion of hospital-to-hospital transmission of research results or information sharing. The only discussion of patient to remote resource use was a lunch conversation I had with a medical assistant and a physician from Alaska who attended the conference specifically because the Internet is being used in such a way in the remote places where health care workers and settlements are few and far between. I didn’t get the opportunity to ask them much about their practice or how the computer-based systems were working out before our lo mein arrived.

Overall, I feel the need to penalize almost all of the speakers for excessive use of cartoons, most of which appeared to be pilfered from *The New Yorker*. Yeah, they get a laugh out of the audience and are sometimes very much to the point, but do we really need over four in a half hour presentation? In addition, almost everyone used slides, even if the materials on the slides were already in the course binder every attendee received.

I learned about the course in Boston by reading the conference announcements in *sci.med.aids*.

---

**The Webmaster:**

**Logging in to your Web Pages**

*by Dave Taylor*

*taylor@netcom.com>*

One area of interest to many people developing Web pages is how to have secure pages that only a subset of the Internet population can access. I’ve seen some silly solutions to this problem, like having a secret port address for the server (you see that when you see URLs that have a format like `http://www.intuitive.com:8054/`, where “8054” is the port address— as opposed to the default port of 80 for `httpd` daemons) or requesting that no one link to the pages on the server. But these techniques are not very effective in the long run.

A much better strategy is to prompt users for a login/password pair, verify that they’re valid, and then validate the users onto the pages on that site. But how to do that? There are three ways: give the user a form to fill in that asks for an account and password, have a pop-up dialog box asking for the same information, or use the built-in mechanisms of your Web server. This time I’ll look at the first and third, and next column I’ll explore the second option.

**Form-Based Verification**

If you’ve been around the net long enough, you’ll remember how *HotWired* and various other Web sites started life requesting that you register with them to get a username for demographic information. Their technology was simple: prompt for the user and password with a button inviting new users to find out how they can be members too. Here’s how the HTML for that form might have looked:

```html
<H1>Please Log In</H1>
<FORM ACTION=apps/verify-login.cgi METHOD=POST>
<pre>
Name: <input type=text name=username>
Passwd: <input type=pass name=password> 
(no account? <a href="no-account.html">Set one up today!</a>) 
<input type=submit value="Sign In">
</pre>
</FORM>
```

The username and password pair would be sent to the CGI script `verify-login` and—if they matched—the user would be let onto the site and allowed access to the files therein.
The script underneath is more complex, but it finally gives us an excuse to look at how to receive and process large streams of data from a Web-based form.

**Receiving Data from a method=post Form**

In this case, the script will need to take the name and password fields from the input stream and compare them with those specified in an existing database (or the actual /etc/passwd file itself). The most complex part of this script is getting data from the input stream when it’s in a method=post format. method=get, you’ll recall, tucks all the data into the query_string environment variable, making it pretty darn easy to work with. POST sends the data from the browser/client as part of the input stream, so the mechanism to read it is a bit more complex; here’s how to do it in C:

```c
unread = atoi(getenv("CONTENT_LENGTH"));
bytesread = 0;
while (unread) {
    if ((retc = read(0, buffer + bytesread, unread))>=0){
        bytesread += retc;
        unread -= retc;
    } else {
        printf("<H1>Error: failed reading CGI stream</H1>\n")
        exit(1);
    }
}
```

The environment variable CONTENT_LENGTH specifies in bytes exactly how much data is in the input stream to read. This input is not carriage return or null terminated. When this code snippet terminates, you have the data stream in the variable buffer.

Once received, the data must be unpacked, however, since a variety of characters are translated into hex equivalents with a % prefix. This translation can be easily done with the following pair of routines—the program need merely call cleanup(buffer) with the information received from the input stream, and the results are sent back in an easily parsed format.

```c
char ^cleanup(string)
char *string;
{
    static char newstring[9999];
    static char tstring[9];
    register int i, j;
    tstring[2]=0;
    for (j=0, i=0; string[i] != 0; i++, j++)
        switch(string[i])
            {
                case '6': newstring[j]="\n";
                break;
                case '+': newstring[j]=’ ‘;
                break;
                case '%': strcpy((tstring,string+i+1,2);
                newstring[j]=atox(tstring);
                i+=2;
                break;
                default: newstring[j]=string[i];
                break;
            }
    newstring[i] = 0;
    return (char *) newstring;
}
```

Now buffer is a sequence of
name=value
name=value
pairs separated by a carriage return (which means that a multiline input area like a <textarea> becomes one long line, among other things).

We’re getting a bit deep in this. Let’s jump back to the main routine of the password checking CGI program and see how things are progressing:

```c
get_datastream(buffer); /* first snippet of code above */
newbuffer = cleanup(buffer); /* unpackage data stream */
/* now let’s get the user & password pair*/
strcpy(username, getvalueof (newbuffer, "username"));
strcpy(password, getvalueof (newbuffer, "password"));
```

The routine getvalueof() whips through the newly cleaned-up buffer and returns the value of the specified name (look at the HTML snippet at the very beginning to see where these two names are coming from).

At this point we’re ready to look the pair up in our database or even search the existing password file for these two:

```c
valid_user(username, password)
char *username, *password;
```
```c
{
    /** returns non-zero if the **/
    /** user + password are in the passwd file **/

    return( pwdauth(username, password)
        == 0 );
}
```

This last snippet of code works only on SunOS systems or other UNIX variants where the system call `pwdauth()` is available. You could substitute any user-password lookup routine for this, of course.

Once the check is done, you can either return an error if it failed:

```c
    printf("HTTP/1.0 403 FORBIDDEN\n");
```

or success (redirecting their browser to the real Web page, in this case, by use of the `Location: http` command):

```c
    printf("Location: %s\n", LOGGED_IN_USER_HOMEPAGE);
```

**Caveat**

Having gone through this long process, I should highlight that most of the more popular Web servers have built-in login authorization capabilities, and that it's pretty darn easy to set things up so that files, directories, or other information are password protected. Typically you'll see this specified in a file called `.htaccess` which might contain a few lines like:

```
AuthUserFile ./passwd/passwords
AuthName "administration"
AuthType Basic
```

Then the passwd directory in that specific directory would contain user and password pairs in a file called `passwords`. A typical entry might look like:

```
nate:zNsIKxG5Exqheion
```

where the password is encoded using, typically, a utility program that came with the Web server itself. Before you go through the hassle of what I list above, please check your server to see if it has these capabilities!

Next issue I'll explore the pop-up login boxes and how to work with them and talk about zones and the handshake of the Basic authorization type.
STANDARDS

An Update on Standards Relevant to USENIX Members

by Nicholas M. Stoughton
USENIX Standards Report Editor
<nick@usenix.org>

IETF GRIP:
Expectations for Security Incident Response

Nevil Brownlee <n.brownlee@auckland.ac.nz> reports on the March 4-8, 1996, IETF meeting in Los Angeles, CA:

The Guidelines and Recommendations for Security Incident Processing (GRIP) working group was formed at the end of 1994 to produce a set of procedures to facilitate the consistent handling of security incidents in the Internet community. Although it is focused on the Internet, many of the concepts discussed in the proposed draft currently available are also useful for other forms of local and wide area network.

The document currently in production is now entitled "Expectations for Security Incident Response" and is available for anyone to read via FTP from your favorite Internet drafts repository (there are several), or

This document is intended to facilitate the setting of expectations regarding the operation of Security Incident Response Teams (SIRTs). It describes the various important topics in the form of a "template," through which every SIRT should describe itself and its functions.

SIRT clients have a legitimate need and right to fully understand the policies and procedures of their Security Incident Response Team. A SIRT's template supplies details for the various important topics that clients must consider when selecting a SIRT.

An example of a SIRT is the Computer Emergency Response Team, CERT, based in Pittsburgh. As the scale of the problem of security attacks increases, so does the number of bodies and organizations offering help. Because many security incidents involve crossing boundaries, whether they are intracompany, intercompany, commercial, national, or whatever boundaries, the handling of such incidents may well involve more than one agency.

In the past, there have been misunderstandings regarding the expectations of these teams. The GRIP guide intends to provide a framework for these expectations and allows the community to express areas and topics that need to be addressed by any SIRT, whatever its specialization.

"Consistent handling" implies that any group calling itself a SIRT must react to security incidents or to threats of them in ways that the Internet community agrees to be in its general interest. Every SIRT needs to clearly define the services it offers and the level at which they are offered to the client. Such definitions will be particularly important in contracts and/or agreements that SIRTs make with their clients.

Errata: Pages 46 and 47 of the May/June issue of ;login: were inadvertently transposed. To the many readers who caught this, and were kind enough to inform us without reverting to epithets, our apologies, especially to our respected authors: Nick Stoughton, Stephen Walli and Jim Isaak.
Our document is now behind our original schedule, but it is beginning to look closer and closer to being a done deal. Probably the next meeting in Montreal at the end of June will see the final review before we submit it to the Internet Engineering Steering Group (IESG) for review.

If this is the sort of area you are interested in collaborating on, please feel free to mail our Working Group chair, Barbara Fraser <byf@cert.org> for more details.

ISO 9899-1:1994: ANSI C

Barry Hedquist <peren!beh@uunet.uu.net> reports on the January 1996 ISO-C meeting in San Jose, CA:

For those of you who have not been watching, you'll be happy to know that the C language we hold so dear is undergoing some changes—all for the better, of course. Amendment 1 to the ISO/IEC C Standard, ISO/IEC 9899-1:1994, was approved late in 1994 and finally got published last year. This amendment is commonly known as the Multibyte Support Extension (MSE).

The MSE defines extensions, primarily to the libraries, that will support multibyte and wide characters, such as kanji, for application programs. This amendment is intended to further promote the international portability of C programs in all environments.

Before anyone gets too excited, it is important to note that none of these changes has an immediate impact on conformance requirements for FIPS 151-2 (POSIX.1) or FIPS 160 (C language) for US Government procurements...at least not yet. Amendment 1 is a separate document from ISO/IEC 9899:1990, and it is not expected to be merged into that document until the work on 'C9X' is completed—yes, more changes are coming.

It is also important to understand that although the terms sound synonymous, there is a distinct difference between “multibyte” and “wide character.” Multibyte refers to a character comprised of more than one byte that is external to an application program, i.e., a kanji character in a file to be read. Wide character refers to the single representation of that character within the application program. When a multibyte character is read by an application, it is converted into a single-representation wide character.

Virtually every library function in ISO C now has a corresponding MSE function, which basically doubles the size of the ISO C libraries. Additional functionality has also been added to a number of the prior I/O functions, such as printf, scanf, etc., as a result of earlier implementations to support multibyte capabilities. New functions related to the conversion of multibyte to wide character and vice versa have also been added.

There are no new trigraphs in ISO C, but we now have six new digraphs to handle those who had problems with using the trigraph representations. There is also a conformance requirement for support of a new header, <iso646.h>, for both freestanding and hosted implementations. This header provides macro definitions for a set of operators.

The extensions made to the C language in Amendment 1 are an important step in improving the utility of this language and broadening its user base within the international community. My next report will address the current plans of the C Standards Committee for C9X.
Books Reviewed in this Column:


Additional Note: Mike Scheer has pointed out to me that the SVR4 file system interfaces stem from the SunOS 4.X VFS/vnode work. He's right, of course.

BOOK REVIEWS

The Bookworm
by Peter H. Salus
<peter@pedant.com>

Summer's here. It's time to do beach or camping reading. I take books on airplane trips. In April, I had to go to Houston for a meeting. I live in Boston. So I did it in one day, leaving at 6:00 am and getting home after midnight, with nine hours in transit. So I took along MBONE and Online Law (both mentioned below). Both flights were pretty full. Outbound, the guy next to me turned out to be a Texas real estate lawyer. He thought that intellectual property law was "crap" because it didn't deal with "real stuff." On the way back, my neighbor was an older woman who told me (looking at the screen dumps in MBONE) that her nephew "knows all that." I assured her that I was trying to learn.

Doing this column gets me the raw material so that I can, indeed, learn.

Freeze-dried Coffee

I received nine more books on Java since my last column. The only one that is really worthwhile is Arnold and Gosling. As a true fan of James Gosling's work on languages since I was exposed to his version of emacs a decade and a half ago, I can only admire what he (and many others at Sun) have done with Java. Ken Arnold's book on ANSI C a few years ago was a keeper, as was his work on GNU emacs. It's an unbeatable combo. No bull. No filler. Real information. Real code. A terse, informative book. What more can I say?

A Big Fish

The next keeper is the second edition of Garfinkel and Spafford's 1991 security book. It has grown immensely, from barely over 500 pages to nearly a thousand, but most of it is both worthwhile and important. Divided into 27 chapters and 7 appendices, G&S proceed from the compulsory history of UNIX (in which they cite me), to the basics, user responsibilities, system security, network and Internet security, "Advanced Topics" (firewalls, wrappers, proxies, etc.), to the handling of security incidents.

If you read the June ;login:, you'll realize that the problems of password and network security are far from new. They date back to the beginnings of both the ARPANET and of UNIX. I recommend Bob Metcalfe's RFC 602 (December 1973 [!]) to everyone.

Security is a really important topic. Your users don't understand it. Educators don't; nor does the medical profession. This is a fine book on a difficult topic. There are lots of things to carp about: I thought the chapter on employees was too brief; I think "Who Do You Trust?" is ungrammatical. Big deal.

This is an important book.

Oh, yes. Henry Spencer's solution to the ORA animal on the cover is still true.

Legal issues

I mentioned reading Online Law in flight, earlier. Online Law is an anthology produced by the Information Technology Law Department of McBride Baker & Coles, edited by Thomas J. Smedinghoff, co-chair of that department. It is a
large, schizoid volume that all of you who are running Web sites, posting on the net, or running network nodes or ISPs should read carefully. It's schizoid because Smedinghoff and his colleagues have tried to write for the layman while supplying the information necessary to legal professionals.

However, if you're willing to plow through the verbiage, this is a very rich field. The sections on rights in a digital environment are thorough and revelatory; those on privacy rights providing the information necessary to legal professionals.

**Multicasting**

One of my friends thought multicasting had to do with fly fishing. It doesn't. MBONE is chock full of ways that the Internet is being used for multimedia communications. The three authors do a fine job of explaining to the interested user just how things are done, how one can do them, what hard environment are thorough and revelatory; those on privacy rights and criminal online conduct are quite fine.

**Coming to Terms**

You can look up MBONE on p. 271 of Shnier's book. You can look up lots of other stuff, too. It's a wonderful and useful book. If it could be combined with the *Telecom Dictionary* it would be as close to perfect as possible. A lot of the telephony stuff is in Shnier's tome-D50, T1, E1, OC—but the explanations are not as thorough or complete as they might be. The "V" section may be the best example of this. Shnier does a fine job of explaining what V.8 and V.17 and V.34bis are. But he never explains that V is the set of ITU-T standards for data communications over a telephone network, nor that "bis" is the second of a series in ITU-T standards or that "ter" is the third in such a series. (ITU and ITU-T are listed.) But this is merely picking and carping; it's another must have.

**Homage to Lee Felsenstein**


Community Memory was a dream that emanated from the Berkeley of the Free Speech Movement and the technology of the PDP-8. It was the first community network. It had a five-page user's manual. The terminals were coin operated: $0.25 to post an opinion. $1.00 to start a new forum. Some of the forums were *Peoples' Park*, *Hacking*, *VDC [Vietnam Day Committee] Reunion*, *Poetry*, *Military Life Facts*, *Senior Cuisine*, *Help Wanted*, and *Help Offered*.

I haven't see Lee in a decade. Community Memory is no more, but there are now hundreds of communication outlets for community and cultural activities.

All this was brought back to me in a nostalgic rush by Shuler's book. Shuler has done a fine job of putting together the history of community networks, their rationale, their concerns, and a lot of how-to information. He also presents several interesting case studies more recent than Community Memory (the Santa Monica PEN project, Cleveland FreeNet, and the Big Sky Telegraph system).

**Anecdotal History**

Hafner and Lyon's *Where Wizards Stay Up Late* is a chatty, entertaining history of the beginnings of the ARPANET, centering about BBN and the IPTO. I had a tough time with it—not just because I have written a similar history, though mine is more technical and covers more territory, but because I feel that there are grave omissions and inaccuracies throughout.

Among the elided is Elmer Shapiro. Shapiro wrote the "Study of Computer Network Design Parameters" at SRI under an ARPA contract in 1968. It was published in December 1968. It supplies the model, based on Wes Clark's suggestion, that was instantiated the next year as the Host-IMP network. Shapiro also wrote RFC 4 (March 1969). But most important, as Steve Crocker noted in RFC 1000, it was Shapiro who insisted on documenting what the NWG did, who is thus the instigator of all the RFCs.

*Where Wizards Stay Up Late* is well-written. But it is repetitious (Herzfeld's girth is mentioned on both p. 10 and p. 41; Wes Clark and the TX0 on both p. 32 and p. 45; Minsky and Papert in several places). Largely because neither Hafner nor Lyon is at all "technical," some things slip through the cracks: Aiken and the ENIAC are there, but not Eckert and Mauchly; Davies and Scantlebury are here, but not Barber. Pouzin and the Europeans are gravely slighted. Norm Abramson and ALOHA finally make their appearance on p. 217. The tale of Metcalfe's dissertation and his invention of Ethernet is retold, but not quite as Metcalfe tells it (in *Packet Communication*, reprinted 1996 by Peer-to-Peer Communications; ISBN 1-57398-033-1). The TCP/OSI tale is so condensed it's meaningless.

Mike Padlipsky's book on OSI is in the bibliography, not in the notes; my book is in the notes, but not in the bibliogra-
TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX® Domain Protocols


Reviewed by Adam S. Moskowitz <adamms@menlo.com>

Let me start by saying that Stevens’s latest book is well written and chock full of information, but you may find that it is dry reading or does not contain the information you seek. Let me say further that this is in no way due to any shortcomings on the part of the author: TCP/IP Illustrated, volume 3, is aimed at implementors of protocol stacks, those who find themselves debugging network problems with a LAN analyzer, or folks writing low-level network code for such things as adaptive packet filters. If you fall into any of these categories, then this is the book for you.

But if you’re looking for a casual discussion of how these protocols work, you are likely to find the line-by-line tours of the kernel sources far more detailed than you need.

One more thing: in addition to the expected table of contents and index, Stevens also provides a rich bibliography, a list of the acronyms used (along with their expansions, of course), as well as an index of data structures, functions, and macros. Although this book can be read on its own, having the two earlier volumes—TCP/IP Illustrated, Volume 1: The Protocols, and TCP/IP Illustrated, Volume 2: The Implementation (with Gary R. Wright)—will prove useful.

The book is divided into three parts: “TCP for Transactions,” “Additional TCP Applications,” and “The UNIX Domain Protocols.” The first of these gets the lion’s share of the book. 12 of the 18 chapters. Stevens starts with an introduction to T/TCP: How it works, why it is needed, examples of how it might be used to improve performance (the best example being HTTP servers), and the nitty-gritty details of what header bits get set and reset, when, and why. He then devotes the second six of the chapters in this section to a function-by-function tour of the kernel sources, with references to the standard implementations (in his earlier volumes) and explanations of how the code is designed to communicate with existing (that is, non-T/TCP) implementations. It is in these chapters that Stevens first points out bugs he found in popular implementations of the TCP protocol. (On a personal note: while I was reading the book, I kept asking myself if the folks at [insert the name of any UNIX vendor] had also read it, and when they were going to fix these bugs.)

Not being a kernel hacker, I found the tour of the sources tough going, but in the end, I was surprised at how much I understood, and just how good a job Stevens did with this admittedly difficult and dry subject matter.

The section subtitled “Additional TCP Applications” covers the Hypertext Transfer Protocol (HTTP), a “day in the life” of a busy HTTP server (at the packet level), and NNTP, the Network News Transfer Protocol. Although the coverage of the first two topics is complete and concise, the third is given a mere ten pages. The information is correct and useful, but I feel it is not tied closely enough to the rest of the book: there is no discussion of how NNTP interacts with the underlying protocols, no mention of T/TCP, and only the briefest table of statistics for a single host. I don’t mean to sound negative, because the chapter is well written, but I feel it belongs in a different book.

In the final section, “The UNIX Domain Protocols,” Stevens once again excels in his description of the protocols and gives yet another thorough and complete tour through the sources. One chapter is devoted to the basic implementation and a second to the potentially confusing topic of descriptor passing. Having read 17 chapters already, I was not surprised to find that the author did an excellent job of keeping things clear, concise, and understandable to “mere mortals.”

Finally, Stevens provides two appendices. The first is a discussion of measuring network round-trip times, complete with charts, boxes and arrows, and several paragraphs explaining each one and how it relates to the code in the rest of the book (a tip o’ the hat to Arlo Guthrie). The second appendix contains a brief but complete discussion of how to write applications that will use T/TCP or TCP, whichever is available. This one chapter is particularly useful to application programmers—those who might have found the discussions of the sources beyond their scope. Also, one does not need to understand the implementation to understand (and use) the tips in this last appendix.

As I wrote earlier, some of this book is dry and slow going, but if you’re implementing protocol stacks, maintaining existing implementations, or need to keep abreast of the latest in what’s happening in your TCP headers, this is definitely the book for you.

IPng: Internet Protocol Next Generation

Reviewed by George Neville-Neil
<gnn@netcom.com>

The current Internet Protocols (IPv4) have been in use for more than a decade. In that time, the Internet has gone from a small, government-funded research network to a worldwide enterprise that is supported by companies, governments, and individuals. The network has grown beyond the expectations that original designers had for it and is now part of the public mind.

All this growth has demonstrated two facts. The first is just how well designed the original protocols were. They have been able to be worked and reworked to handle nearly exponential growth for many years. The second is that the current protocols need to be reworked if many more people are to be connected to the net. Though a 32-bit address seems sufficient, the way it has been partitioned is not efficient enough to allow for the assignment of many more addresses. Furthermore, the routing mechanisms of the current Internet cannot handle much more expansion.

The Internet Engineering Task Force has known of these problems for some time and began—in 1991—setting up various working groups to deal with them. A new protocol, called IPv6, has grown out of the work of various groups since 1991. This book attempts to give an idea of what went into this design process and then presents the results.

The book is broken down into several parts. The first three parts discuss the need for a new protocol, describes the process that was used to come up with it, and estimates how long the current protocols would last before all the addresses were exhausted and/or the routing tables became completely unmanageable.

Part IV is a long section that collects the papers that were written in response to Request For Comments (RFC) 1550. RFC 1550 "IP: Next Generation (IPng) White Paper Solicitation," was a call for papers about requirements and selection criteria from various interested groups (e.g., industry, government, the military). The papers in this section cover a broad range of topics, from military simulation to the use of IPng in the cable and wireless network industries.

The next part is a more technical discussion of the issues facing IPng and how they might be solved. This section covers such issues as security, addressing, and transition between the old and new versions of the Internet Protocol.

Part VI presents the technical criteria used in selecting the new protocol. Part VI presents the three proposals that were submitted and judged, as well as a justification for the final decision.

Part VIII is a description of the final recommendation, IPv6, including a technical overview, a brief discussion of address autoconfiguration, and a lengthy section on transitioning from IPv4 to IPv6.

The last two sections of the book cover security in the new protocol and the ongoing process of revising the protocols. An appendix and index conclude the book.

When I sat down to read this book, I thought, "Isn't it a bit early to be writing a book on a protocol that's barely been deployed?" It was several years before Douglas Comer's book on TCP/IP was published, and several more before Stevens and Wright published an illustrated book on the implementation of the protocol.

Unfortunately, I was partially right. As a description of the new IP protocol, this book has been written too early. It is more of a description of the process under which IPng was developed. Part IV is filled with papers that say almost the same thing as each other, only in different ways. Everyone wants more addresses, quality-of-service mechanisms, and security. This section could have been much shorter, but was probably kept the way it is to include all who had contributed to the process. Of the 300 pages of text in this book, almost one-third (91 pages) are dedicated to the white papers.

After getting through part IV, I felt that the more technical part V would be a respite. Unfortunately, this section simply repeated much of what was said in part IV, but this time from the perspective of those who were going to write the new protocols. This section, a further 48 pages, added to what I would consider the nontechnical content of this book, bringing the total to about half.

There were further problems in that the book is not well edited. The most glaring example of this is on page 185, where the end of the second paragraph and the fifth paragraph are nearly identical.

Having read the book, as well as the RFCs themselves, I think that someone interested in the mechanisms IPng protocols would not be interested in this book. But if you want to know why certain decisions were made in the IPng process, then this is the book for you.
IPng: Internet Protocol Next Generation


Reviewed by Steve Simmons
<scs@lokkur.dexter.mi.us>

It's often said that there are no failures in science because we learn even from the experiments that fail. This is true only if those failures are remembered, so that future researchers can avoid the things already shown not to work. If there is no record, then we will likely run down the same blind alley again and again.

This principle also applies to the development of standards and applications. When examining a program or a network protocol, it is difficult to determine which design decisions were random selection among near-equal choices, which were mandated from above, and which were made in order to avoid problems known only to the original developers. Our ignorance of the history can lead us to make inappropriate or incorrect criticisms of a work, while excess deference to the original developers has the potential to make us excessively timid about attempting to remedy a problem.

I've been working with TCP/IP for a little over a decade. Along the way I've often wondered why particular implementations were designed the way they were, why some features were present and not others, and what process enabled those decisions to be made. Lacking a reliable history of the development process, it's been impossible to understand, excuse, or justify certain decisions.

Thanks to the efforts of Scott Bradner and Allison Mankin, this will not be a problem with IPng. Their new book, IPng, Internet Protocol Next Generation, covers exactly these topics. If you're looking for a book that will do for IPng what Comer or Stevens do for IP version four, this does not fill the bill. In fact, given that IPng has barely entered the implementation stage, it's grossly premature to expect such a book. But if you want to know the criteria used in deciding what would be used and why, this is the book for you.

Bradner and Mankin are co-chairs of the IETF IPng process, and as such are intimate with the entire process of developing the protocols. The book starts with a discussion of the problems, then proceeds through the gathering of requirements, the evolution of competing proposals, and concludes with the implementation and transition plans. It may seem odd to call something a history when the protocol under discussion hasn't been fully implemented yet; Bradner and Mankin have written a history not of IPng's implementations but of its design.

The book comprises a series of essays by various authors. This diversity of voice is an asset in the early parts of the book, particularly part IV, "The Role of IPng in Communications Technology," and part V, "Features, Technologies, and Issues for IPng." Each author presents his own personal or corporate view on what is needed in IPng and why, thus providing a lively and sometimes contradictory presentation of needs and features. In a few of the articles, one could have wished for a more active editorial hand; one author uses confusing or odd numbers like 16^9 (yes, sixteen to the ninth power) and 4^3 rather than the more accessible 68 billion and 26 respectively.

As the book proceeds toward the decisions on requirements and implementation, the articles become clearer, longer, and more coherent. It's clear that while complete consensus did not emerge, there was a realization that a decision had to be made and that the choices were among alternatives which worked.

Is this book something you would use to learn the new protocols? Definitely not. The authors refer you, quite correctly, to the RFCs. Because implementation is ongoing, there is a significant chance that any book written on IPng itself will be out of date on the day it is published. But as a history of a technological process, this book is immensely useful and will no doubt be required reading when the next set of researchers addresses the next protocol revision.

Hands-On Netscape: A Tutorial for Windows Users


Reviewed by George W. Leach
<gwll@gte.com>

Hands-On Netscape provides an end user with a tutorial introduction to Web surfing using Netscape Navigator on a PC running MS Windows. Each chapter includes several tutorial sessions and "video tutorials" or movies to assist the reader. The movies are contained on a CD that comes with the book. Also contained on the CD are some helper applications, including Adobe Acrobat Reader 2.0, QuickTime 2.0 for Windows, and a shareware MPEG player and audio player.

The material is partitioned into three basic sections. "Netscape Ready" covers issues such as what is Netscape, getting started with the Internet, installing or downloading Netscape, and an introduction to the Web. "Netscape Set" provides the reader with a quick tour of the user interface features of Navigator, discusses URLs, investigates what's out there on the Web, the history mechanism, bookmarks, setting options, and a tour of the Web. Part three, "Netscape Go!" covers search engines, using Navigator for non-Web access to Internet services (Telnet, FTP, gopher, email, and USENET), multimedia (PC speaker drivers, GIF and JPEG
images, Adobe Acrobat Reader, and QuickTime), a brief introduction to HTML, and customizing Navigator.

A list of Internet access providers is contained in an appendix. Other appendices provide a brief introduction to TCP/IP and information on downloading win32s software, which is necessary for running some of the helper applications on the CD unless the reader is using Windows 95.

Well over 200 figures, most of which are screen shots, are provided to assist the reader in learning how to use Netscape Navigator and to illustrate some interesting Web sites. Unfortunately, much of this material is out of date. Many screen shots depict the Netscape home page and others, that have changed considerably since this book was printed. The book was published prior to versions of Netscape Navigator being available for Windows 95 or Windows NT, therefore these are not covered.

This book is not for your average USENIX member, but it does make an excellent tutorial for the typical office worker who is somewhat computer literate. The level of feature coverage is very basic, but that is OK unless someone intends to become a power user of Netscape. So if you have an office full of folks to train, this book just might do the trick, although it would be useful if the authors kept the book up to date with new releases of Navigator.


UNIX Tamed
Reviewed by Rick Umali
<rgu@world.std.com>

Can one “quickly master” UNIX? That’s what the cover of UNIX Tamed, by Rodney Wilson, claims. However, reading the preface reveals that this book is “recommended as a supplement to most introductory or system administrator courses” for UNIX. In this modest goal, the book succeeds.

Each chapter is a blend of explanation followed by exercises. (The answers are at the end of the book.) By trying the exercises, and doing careful independent reading of the manual pages, one can quickly dive into the many facets of UNIX.

The first four chapters (part of part one, “Getting Started”) introduce the reader to logging in, logging out, and the most common UNIX commands (pwd, cd, ls, cp, rm, mkdir, grep). The reader is then brought a little deeper into the waters that are UNIX (umask, meta characters, environment variables, the tee command, links). The reader is certainly bound to gain much benefit by trying the exercises, and following the examples (yes, the book is certainly meant to be read by the terminal).

Chapter 5 (which begins part two, “Moving Ahead”) introduces the user to ex and vi, with a little teaser on the Korn shell. By the end of chapter 6 (the Bourne shell) and chapter 7 (the C shell), the reader is (ideally) writing some simple scripts. Further, the reader gets a taste of the power of these shells. For me, going over the examples showed me a lot about CSH that I never knew or had forgotten.

Part three (“Getting Good”) is a true hodgepodge of topics. Chapter 8 describes sort, find, sed, and awk. This chapter is touted as advanced UNIX commands. Chapter 9 is a shallow dive (nine pages!) into the facilities that help programmers (make, RCS, and ar). As a non-system administrator and a non-Perl hacker, I liked the skimming of both these topics (covered in chapters 10 and 11).

After finishing the book, I felt as if I was taken through a really quick tour of a museum full of intricate exhibits and pieces. Now that I’ve seen everything, I want to go back and take some more time.

I had a problem with the typeface selection. Using a proportional typeface for computer output made some of the examples confusing. I also had a problem with some of the ordering. Case sensitivity is mentioned on page 89 in a footnote. Email (mailx) is used in a few examples, but it probably should be left out. Still, UNIX is so big that coherently trying to describe it without forward references is difficult.

I believe the best way to learn UNIX is “the old-fashioned way”: manual pages, experimentation, and learning from other experts. Rodney’s book is good for pure beginners because he introduces the reader to the experimentation that is so necessary to learn UNIX.

I’m not sure if UNIX is tamed in the 160 pages (minus appendix and index) that is Rodney’s book. His book provides a fine skimming of the vast ocean that is UNIX. With occasional dives into the water (some deep, some shallow), the reader can quickly see and learn a lot about UNIX. Trust the author and purchase his book with other more “intensive” books to truly tame the UNIX beast.


Reviewed by David L. Oppenheimer
davido@cs.princeton.edu

*Information Security Policies Made Easy* is not a book for individuals lacking prior experience formulating information security policies. It is, however, an excellent compendium of security policy statements for those who have a general idea of the overall policy they want but may not know precisely how to translate their information security goals into concrete words.

This 426-page text will not solve the difficult institutional dilemmas which managers must confront when writing information security policies. And because it is "a comprehensive set of information security policies" rather than a step-by-step guide to formulating and writing such policies, this text will be much more useful to a corporate information security manager than to the average system administrator who wishes to draw up guidelines for usage of the system(s) he or she maintains.

*Information Security Policies Made Easy* is structured as a categorized list of policies, each one a few sentences long and followed by a paragraph of generally enlightening commentary. Wood annotates each policy with an indication of the security environment to which he feels it applies, in particular a "low-security," "medium-security," or "high-security" situation. This categorization is of dubious value, however; anyone put in a position of writing an information security policy would certainly be expected to understand which policies apply to the environment of their organization and which do not. Moreover, some of the author's categorizations seem a bit odd. For example, a policy requiring that writable media loaned to a company, used in that company's computers, and then returned to the loaner be destroyed instead of returned, is considered applicable only to "high security" environments. Yet a rule forbidding the use of any software not centrally administered through a company-wide licensing/distribution scheme is said to apply to low, medium, and high-security environments. The first rule would seem applicable to a wider audience than the author suggests, while the second seems a bit excessive for low-security concerns.

*Information Security Policies Made Easy* is truly a book about information security, not just computer security. It's guidelines range from policies mandating the segregation of development machines from production machines, to guidelines designed to prevent PBX fraud, to suggestions for maintaining physical security of a company's tangible and intellectual property. The author appears to have a broad grasp of the important issues in the management of information security; Wood's paragraphs of commentary following each policy bring out important points not always obvious from the policies themselves. He is able to keep these commentaries brief by minimizing their technical content; this seems a wise decision because the book is targeted to organizations which may use any type of information processing equipment, not just to networks of UNIX systems.

But these commentaries cannot replace the need for a technically skilled individual with the ability to distinguish useful policies from useless ones, to be given the responsibility of writing a company's information security policy. First, many of the suggested policies in Wood's book are simply impossible to implement using commodity hardware and software systems used in most non-governmental environments today. And even if one were to develop a corporate information system that enabled implementation of all of the listed policies, implementing all of them would almost certainly yield a system which prevented its users from doing much work apart from that directly related to upholding the corporate security policy. Second, as the author notes, some of the policies contradict one another since the book tries to appeal to a wide range of security mentalities. For example, one policy mandates the use of "single sign-on" while another requires that users employ a different password for each system they access. Thus only an individual with sufficient technological background to understand the meanings of the policies in this text will be able to make productive use of the book.

*Information Security Policies Made Easy* is a comprehensive information security policy but which desire some suggestions on how to formulate the specifics of that policy. The text deals with almost all conceivable facets of information security, making it a great aid for writing a high-level corporate information security policy. On the other hand, *Information Security Policies Made Easy* is probably overkill for system administrators who are simply trying to write a list of guidelines for their users. Individuals in the latter category would be better advised to search the Internet for sample policies or to look for a book which deals solely with computer security policies. And with a price tag of $495 for an "organizational-wide license," *Information Security Policies Made Easy* isn't the sort of book a company is likely to buy unless it has a fairly strong commitment to establishing an organization-wide information security policy or upgrading an existing one.

*Information Security Policies Made Easy* comes with a disk which presumably contains the text of the policies contained within the book. This disk was not supplied to the reviewer so we cannot comment on it.
Call for Participation: HotOS-VI
Sixth Workshop on Hot Topics in Operating Systems
May 5-6, 1997
The Wequasset Inn - Chatham (Cape Cod), Massachusetts
Sponsored by the
IEEE Computer Society Technical Committee on Operating Systems and Application Environments

The Sixth Workshop on Hot Topics in Operating Systems will bring together people who write applications and have ideas about what an operating system should provide, people who build operating systems for a living and care how they are put together, and researchers in operating systems who think they know how to solve both kinds of problems. The purpose of HotOS-VI is to explore the goals, costs, and compromises of operating system design and implementation, and to encourage discussion of novel, controversial, or unjustly abandoned ideas in the field. The workshop is designed to encourage full participation of each attendee; both presenters and audience will be active contributors throughout the workshop.

We would like to explore questions such as:

- What do applications need from operating systems?
- What do users need from operating systems?
- How can we shorten the operating system product development cycle?
- How do we support very large and very small systems?
- How can users avoid reference manuals and system administrators? Can our systems self-configure, self-organize, and self-repair?
- How do we support databases? Internet services? Games?
- How do we support systems with very slow networks? With very fast networks? With insecure networks?

To assure a productive workshop environment, attendance will be limited to 60 participants active in the field. Each potential participant should submit ten copies of a position paper of no more than 3000 words (not counting references). We will favor position papers that propose new directions, advocate non-traditional approaches, or generate controversy and discussion. We encourage submissions from practitioners as well as from researchers. Please do not submit abbreviated versions of conference or journal papers.

Position statements will be distributed to participants before the workshop, and will appear in a proceedings volume. The program committee may invite the authors of the best position statements to provide an extended version of their statements for the workshop proceedings.

Submission deadline: January 15, 1997
Notification of acceptance: February 28, 1997
Final position papers due: March 28, 1997

General Chair: Brad Chen, Harvard University
Program Chair: Jeffrey Mogul, Digital Equipment Corp. Western Research Lab.
Publicity Chair: Puneet Kumar, Digital Equipment Corp. Systems Research Ctr.
Finance/Registration Chair: Joseph Boykin, CLARiiion Advanced Storage Solutions

Program Committee:
Mary Baker, Stanford University
Pei Cao, University of Wisconsin
John Carter, University of Utah
Peter Druschel, Rice University
Rob Gingell, Sun Microsystems
Bruce Lindsay, IBM Almaden Research Center
Rob Pike, Bell Laboratories
Jerome H. Saltzer, M.I.T.

Steering Committee:
Brian Bershad, University of Washington
Joseph Boykin, CLARiiion Adv. Storage Solutions
Fred Dougalis, AT&T Research
Mike Jones, Microsoft Research
Darrell Long, University of California, Santa Cruz

Send submissions (10 printed copies) to:
Jeffrey Mogul/HotOS-VI Program Chair
Digital Equipment Corp. Western Research Lab.
250 University Avenue
Palo Alto, CA 94301

Send questions to hotos6-info@pa.dec.com

Financial support from:
AT&T, Digital Equipment Corporation, Lucent Technologies, Microsoft, and Sun Microsystems
ANNOUNCEMENTS & CALLS

10th Systems Administration Conference (LISA '96)

September 29 - October 4, 1996
Chicago, Illinois

Tutorial Program:
Sunday, September 29

Joining the Internet Safely Using UNIX and Firewalls
Introduction to UNIX System Administration
System and Network Performance Tuning
Expect - Automating Interactive Applications
Introduction to HTML
Secure System Administration with Kerberos
Introduction to DNS and Bind
Advanced HTML Design
Your Legal Rights and Liabilities as a System Administrator
Advanced Topics in DNS and BIND

Monday, September 30

NIS+
New Topics in System Administration (part 1)
IP Version 6: An Introduction
Beginning Perl Programming for UNIX Programmers
   (updated for Perl 5)
Setting Up and Administering A Web Server
IP Network Administration
Obscenity, Indecency and the Net: Your Responsibilities as a
   System Administrator
Talking Technical - Breaking the Communication Barrier
Sendmail From The Trenches
Applied JavaScript
Effective Meetings: Get More Done In Less Time
What's New in Sendmail 8.8

Tuesday, October 1

Connecting to the Internet
Sendmail Inside and Out (updated for Sendmail 8.8)
Internet Security for System and Network Administrators
Selected Topics in System Administration (part 2)
Solaris System Administration
CGI and WWW Programming in Perl
Security of the World Wide Web
Administering the Network Information Service: Making
   NIS Work For You
Introduction to NNTP and INN
Administration of MS Windows NT Server 3.51
TCP/IP Troubleshooting with UNIX
Advanced Topics in NNTP and INN
Writing Good Stuff: A Practical Guide for Technical Content

Workshop:
Advanced Topics in System Administration
Tuesday, Oct 1, 1996

A one-day workshop organized by John Schimmel of Silicon Graphics will focus on the latest-breaking technical issues in the systems administration arena. Attendance is limited and based on acceptance of a position paper.

How to submit: Potential workshop attendees are invited to submit a proposal by early August. Contact John Schimmel via email at <jes@sgi.com> for more information. Acceptance notices to all participants will be issued by September 9, 1996. Participants must be pre-registered for the LISA conference. No additional fee will be charged to attend this workshop, and lunch will be provided.

Technical Sessions:
Wednesday, October 2

Keynote Address: Information Technology:
The Next Ten Years
Dick Lampman, Hewlett-Packard Company

Refereed Papers:

Security:

Priv: Secure and Flexible Privileged Access Dissemination
Brian C. Hill, University of California, Davis

The Igor System Administration Tool
Clinton Pierce & John Bell, Ford Systems Integration Center

Centralized Administration of Distributed Firewalls
Mark Miller and Joe Morris, Bell Atlantic

Account Management:

Shuse: Multi-Host Account Administration
Henry Spencer, SP Systems

The Design and Implementation of a Network Account Management System
J. Archer Harris, & Gregory Gingerich,
James Madison University

UNIX Host Administration in a Heterogeneous Distributed Computing Environment
Gregory S. Thomas, Desiree C. Johnson, John P. Moore,
Merrilee E. Orcutt, James O. Schroeder, & Jeffrey T. Simmelink, Pacific Northwest National Laboratory
ANNOUNCEMENTS & CALLS

Visualization in Systems Administration:
Visualizing Huge Tracefiles with Xscal
Alva L. Couch, Tufts University

Using Visualization in System and Network Administration
Doug Hughes, Auburn University

Invited Talks

Standards—Are They Worth The Effort?
Moderator: Rik Farrow, Internet Security Consulting
Panelists: Nick Stoughton, PERT Systems Ltd.; Louis Imershein, Santa Cruz Operation; Lee Damon, Qualcomm

Scaling Your Web Server—What to Do With a Million Hits Each Day
Dan Klein, LoneWolf Systems

ATM: Not Just A Type of Bank Machine Anymore
Peter Van Epp, Simon Fraser University

Thursday, October 3

Refereed Papers

Tools:

How to Avoid Learning Expect—or—Automating Interactive Programs
Don Libes, NIST

An LPD for the 90s
Mark H. Fletcher, SAS Institute Inc.

RUST: Managing Problem Reports and To-Do Lists
Craig R. Ruefenacht, University of Utah

Networking:

Renumbering: Threat or Menace?
Eliot Lear, Jeff Coffin, Rod Scott, Jennifer Katinsky, Diane Tharp, & John Parisi, Silicon Graphics, Inc.

OC3MON: Flexible, Affordable, High Performance Statistics Collection
K. Claffy, NLANR/UCSD; Joel Apisdorf, & Rick Wilde, MCI

IP Multiplexing by Transparent Port-Address Translation
Heon Y. Yeom, Jungsoo Ha, & Ilhwan Kim, Seoul National University

Sendmail:

Many Mail Domains, One Machine: The Forwarding Mailer
Hal Pomeranz, NetMarket/CUC International

How to Get There From Here:
Scaling the Enterprise-Wide Mail Infrastructure
Michael Grubb, Duke University

Automatic and Reliable Elimination of Email Loops Based on Statistical Analysis
Eduardo Solana, V. Baggiolini, M. Ramluckun, & J. Harms, Université de Genève

Toasty Cool Moose:

MajorCool
Bill Houle, NCR Corporation

The PGP Moose—Implementation and Experience
Greg Rose, Qualcomm International

The Brave Little Toaster Meets Usenet
Karl L. Swartz, Stanford Linear Accelerator Center

Invited Talks

How to Run a Worldwide Network
When You Work in the Center of the Universe
Joel Avery, Nortel

What It’s Like to Be Your Own Boss
Tina Darmohray, Consultant & Celeste Stokely, Stokely Consulting

Experiences of Running a Large Archive Site
Stuart McRobert, Imperial College, London

Works-In-Progress Reports

To be announced.

Friday, October 4

Refereed Papers

Software Distribution:

A Simple Caching Filesystem for Application Serving
John D. Bell, Ford Systems Integration Center

Automating the Administration of Heterogeneous LANs
Michael Fisk, New Mexico Institute of Mining and Technology

PC Administration Tools: Using Linux to Manage Personal Computers
Jim Trocki, American Cyanamid Company

Abstract Yourself with Modules
John L. Furlani, SunSoft, Inc. & Peter W. Osel, Siemens AG
ANNOUNCEMENTS & CALLS

SLINK: Simple, Effective Filesystem Maintenance Abstractions for Community-Based Administration
Alva L. Couch, Tufts University

Managing and Distributing Application Software
I. Reguero, Ph. Defert, E. Fernandez, M. Goossens, O. Le Moigne, & A. Peyrat, CERN, European Laboratory for Particle Physics

The Future of Systems Administration:
A New Twist on Teaching System Administration
Raven Tompkins, Indiana University

Institute White Pages as a Sys Admin Problem
Jon Finke, Rensselaer Polytechnic Institute

New Fangled Phone Systems
Pose New Challenges for System Administrators
Snoopy, iXOS Software GmbH

Invited Talks
Manage People, Not Logins
Jon Finke, Rensselaer Polytechnic Institute

Intrusion Detection
Louis Todd Heberlein, University of California, Davis

Just Another Convicted Perl Hacker
Randal Schwartz, Stonehenge Consulting Services

Closing Session
System Administration: The Last Ten Years and the Next
Rob Kolstad, Berkeley Software Designs, Inc.

Student Stipends
The USENIX student stipend program covers travel, living expenses, and registration fees to enable full-time students to attend USENIX meetings. Detailed information about applying for a stipend is available at the USENIX web site: <http://www.usenix.org>, by reading comp.org.usenix or sending email to <students@usenix.org>

Birds-of-a-Feather Sessions (BoFS)
Tuesday, Wednesday, and Thursday evenings. Do you have a topic that you’d like to discuss with others? BoFS are very interactive and informal gatherings for attendees interested in a particular topic. Schedule your BoF in advance by sending email to <conference@usenix.org> or by telephoning the USENIX Conference Office at 714.588.8649. BoFs may also be scheduled on-site at the registration desk.

The Guru is In
Have a question that’s been bothering you? Experts from the USENIX community will be available to spark controversy and answer questions. These are informal discussions among participants, one more way at the conference to transmit information. Please contact Steve Simmons via email to <scs@lokkur.dexter.mi.us> if you would like to volunteer your expertise.

Terminal Room
The Terminal Room will provide Internet and dial-out access, along with laptop facilities. PPP access from your Marriott Hotel room will also be available. The Terminal Room will be open Monday - Friday.

Would you like to become a Terminal Room volunteer? Terminal Room volunteers receive a complimentary technical sessions registration. Look for details posted to comp.org.usenix.

Works-in-Progress Reports
Short, pithy, and fun. Works-in-Progress Reports (WIPs) 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 Adam Moskowitz via email to <lisawips@usenix.org>. A list of topics is announced on-site.

Additional Information
To obtain full descriptions concerning the tutorials, technical sessions, discounts, registration fees, and a registration form, visit the USENIX Web site: http://www.usenix.org or send email to <conference@usenix.org>
Vendor Exhibits at LISA '96,
October 2-3, 1996

80 Vendors of innovative systems administration and network management solutions will demonstrate their products at LISA '96.

Exhibiting companies as of July 3, 1996:

- AIM Technology
- AT&T CommVault Systems
- Auspex Systems, Inc.
- Bay Networks, Inc.
- Boole & Babbage, Inc.
- Border Network Technologies, Inc.
- Central Data Corporation
- Central Design Systems Inc.
- Clarinet Communications Inc.
- Clarity Software Inc.
- Competitive Automation >Cray Research, Inc.
- CrossWind Technologies, Inc.
- Cypress Consulting, Inc.
- DataLynx, Inc.
- Devcom Mid America Inc.
- Digital Equipment Corporation
- Einstein's Universe
- Elegant Communications Inc.
- ENlighten Software
- Enterprise Systems Mgmt Corp.
- Falcon Systems Inc.
- Fastlane Systems Ltd.
- FSA Corporation
- Fujitsu Microelectronics Inc.
- GraphOn Corporation
- Internet Security Systems, Inc.
- Landmark Systems Corporation
- Legato Systems, Inc.
- LSC Incorporated
- Miller-Freeman, Inc.
- Network Appliance, Inc.
- O'Reilly & Associates, Inc.
- Open Systems Management Inc.
- Paranet
- Parity Systems, Inc.
- PDC
- Pencom Systems Inc.
- Personal Productivity Tools, Inc.
- Platinum Technology, Inc.
- Prentice Hall PTR
- QMASTER Software Solutions Ltd.
- RDI Computer Corporation
- SCH Technologies
- Shiva Corporation
- Softlink (USA), Inc.>
- Software Moguls
- Spectra Logic Inc.
- Storage Computer Corporation
- SunSoft, Inc.
- SyncSort Incorporated
- Taos Mountain
- TeamQuest Corporation
- Transarc Corporation
- Trusted Information Systems, Inc.
- Underscore, Inc.
- Unisolutions Associates
- Walnut Creek CDROM Inc.
- Workstation Solutions, Inc.

For more information about the vendors exhibits, please contact: Cynthia Deno, Exhibition Coordinator: Phone: 408.335.9445; Email: <display@usenix.org>.
Announcing

Second USENIX Workshop on Electronic Commerce

November 18–20, 1996
Claremont Hotel & Resort, Oakland, CA

Sponsored by the USENIX Association
Program Chair: Doug Tygar, Carnegie Mellon University

Program Committee (partial)
Ross Anderson, Cambridge University
Nathaniel Borenstein, First Virtual Holdings, Inc.
Stefan Brands, CWI
Dan Geer, Open Market, Inc.
Clifford Neuman, University of Southern California
Hal Varian, University of California, Berkeley
Bennet Yee, University of California, San Diego

Overview
The Second USENIX Workshop on Electronic Commerce will provide a major opportunity for researchers, experimenters, and practitioners in this rapidly self-defining field to exchange ideas and present results of their work. This meeting will set the technical agenda for work in the area of Electronic Commerce by examining urgent questions, discovering directions in which answers might be pursued, and revealing cross-connections that otherwise might go unnoticed.

Tutorials
These tutorials will focus on:
- Getting paid on the internet
- Contracts, Records, and Privacy
- Secure Java Programming
- Securing the Web
- New Java Programming Tools

Plus the latest research on: EC systems that are up and working, Economic studies, Tools for building systems, Security analyses, Case studies and Holes discovered in certain commerce systems.

Workshop Topics
Two days of technical sessions will follow the tutorials. Birds-of-a-Feather sessions in the evenings and a keynote speaker will round out the program.

Among the wide range of issues and ongoing developments to be discussed are:
- Advertising
- Anonymous transactions
- Auditablety
- Copy protection
- Credit/Debit/Cash models
- Cryptographic security
- Customer service
- Digital money
- E-mail enabled business
- EDI
- Electronic libraries
- Electronic wallets
- Exception handling
- Identity verification
- Key management
- Legal and policy issues
- Micro-transactions
- Negotiations
- Protocols
- Reliability
- Rights management
- Service guarantees
- Smart-cards

Proceedings
Published proceedings will be available free to all Technical Workshop attendees, and will be available for purchase at the conference registration desk.

Registration Information
Materials containing all details of the technical and tutorial programs, registration fees and forms and hotel information will be available September 1, 1996. If you wish to receive the registration materials, please contact USENIX at:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630
Tel: 714.588.8649 Fax: 714.588.9706
Email: conference@usenix.org
URL: http://www.usenix.org
Call for Proposals and Presenters

The UniForum Association invites presenters to submit proposals for its annual conference in the following general areas of Unix and open technologies:

- Operating Systems - Hardware and Software
- The Internet, the Intranet, the World Wide Web
- Application Development and Tools
- Open Network Computing
- Client/Server, Middleware and Legacy System Migration
- Data Base, Data Mining, Data Warehousing Implementation Strategies
- PC and Unix Integration
- Systems and Network Administration & Management
- Computer Telephony Integration
- Security
- Emerging and Advanced Technologies

One and Two-day Conference Workshops, Tutorials and Seminars will take place March 10-11, while Track Sessions and BOFs will run March 12-14.

What Your Proposal Should Include

Your Proposal should identify the type of session you are interested in giving: multi-day or single-day Workshop, Tutorial or Seminar; Track Session or BOF.

Your Proposal should include a paragraph description about the technology to be discussed, its open technology connection, and a problem/solution case statement that speaks to how the technology works in the enterprise. A target audience should be named with pre-requisites, if any. A brief presenter's background should give pertinent information on expertise and conference experience.

How to Submit Proposals

The preferred method is to send your Proposal via email (ASCII) to: conference97@uniforum.org

Or by mail to: Claudia Marshall, Conference Coordinator, UniForum, 2901 Tasman Drive, Suite 205, Santa Clara, CA 95054. Your submission will be acknowledged.

About the UniForum '97 Conference

The UniForum Conference is the largest educational event of its kind for the practitioner in the enterprise who is working with, or plans to work with, Unix systems and open technologies. The Conference attendee will have a computer systems background, but not necessarily Unix systems experience. The attendee will be looking for substantive, practical information that can be applied on the job or used for strategic planning and/or procurement.

Proposals for UniForum '97 Conference sessions should target this audience. Proposals that are case study oriented and which call for participative feedback are preferred.

For more information go to the UniForum home page on the World Wide Web:
http://www.uniforum.org or send inquiries to conferences@uniforum.org
USENIX 97 Exhibits

At the USENIX Annual Technical and USELINUX Conferences

January 8-9, 1997, Anaheim Marriott Hotel, Anaheim, CA

“A major gathering for the UNIX tribe, (USENIX) focuses on the latest technology and techniques that can be applied immediately.” Hot Happening, ComputerWorld, 11/20/95

“USENIX meetings are still attended by the breaking edge people in software and systems, but are informal enough that the novices can meet and talk with the more experienced. The meetings have increased in size and have become more diverse, but are still fun, thought provoking, and above all practical.” Steve Johnson, :login; 6/96

Demonstrate your application development, programming, network management or system administration products and services to the most technically knowledgeable group in computing—USENIX UNIX USERS.

USENIX attendees are sophisticated programmers, developers, system administrators, network managers, engineers, and researchers. When surveyed, they tell us that they are working on, supporting, and developing for many different UNIX and other-than-UNIX platforms. They use UNIX on a daily basis and are committed to the newest tools and technology available. At Anaheim, we are conservatively predicting a gathering of 2000 advanced computing professionals—all of whom are committed to the newest tools and technologies on display in the Exhibit Hall.

“Two days of exposure to the cream of the UNIX User Community.” Neil Groundwater, Enterprise Management Group, SunSoft, Inc.

“My competitors aren’t here, and they don’t know what they’re missing.” Brian Duggleby, UNIX Marketing, Digital Equipment Corp.

Companies with reserved space at USENIX ‘97:

- Addison-Wesley
- Advanced Digital Information Corp.
- Atria Software Inc.
- Auspex Systems, Inc.
- Boole & Babbage
- Central Design Systems Inc
- CrossWind Technologies, Inc.
- Digital Equipment Corp.
- Enhanced Software Technologies Inc.
- Enterprise Systems Management Corp.
- Falcon Systems Inc.
- FSA Corporation
- GraphOn Corporation
- Lucent Technologies, Inc.
- Max Enhancement Group
- Miller-Freeman, Inc.
- Network Appliance, Inc.
- O’Reilly & Associates, Inc.
- Parity Systems, Inc.
- PDC
- Prentice Hall PTR
- Raima Corporation
- RDI Computer Corp.
- San Diego Technical Books
- Specialized Systems Consultants, Inc.
- Storage Computer Corporation
- SunSoft, Inc.
- Transitional Technology Inc.
- Walnut Creek CDROM Inc.
- Western Scientific
- WRQ, Makers of Reflection) Software

Contact Cynthia Deno to reserve exhibit space: 408.335.9445 Email: <display@usenix.org>
Do You Want to Expand Your Linux Market?

If you are a software applications developer, OEM, VAR, or Distributor interested in these topics:

- the Linux market: Who, What, When, Where and Why?
- Application portfolios: What's available, what can be done
- Marketing to the Linux marketplace
- Channels for selling your applications and services
- Licenses and licensing
- Porting Linux applications
- Linux system administration
- Developing the Linux operating system and environment

then plan to attend the USELINUX Conference, co-located with the USENIX 1997 Annual Technical Conference.

Attend both conferences for the same fee.

For more information:
Mail: USENIX Conference Office, 22672 Lambert Street, Suite 613, Lake Forest, CA 92630
Phone: 714.588.8649
Fax: 714.588.9706
Email: conference@usenix.org
WWW: http://www.usenix.org

Register Early and Save $100!
Mastery a topic means going beyond the mechanics and gaining a thorough understanding of the problem you’re trying to solve. It’s this kind of thinking that’s behind all our books; you’ll gain a “big picture” perspective and will learn, at the same time, time-saving, practical tips and tricks the experts use.
Please study all orders to Wiley & Sons, Inc.:

Attn: Karen Cooper, Special Sales
605 Third Avenue
New York, NY 10158
Phone: (212) 850-6789
Fax: (212) 850-6142

The Internet Navigator, 2nd
Paul Gilster
1-05260-4 $24.95 member price: $21.20
# of copies: __________

Advanced Topics in UNIX
Ronald Leach
1-03663-3 $24.95 member price: $21.20
# of copies: __________

Introduction to Client Server Systems
Paul Renaud
1-57774-X $34.95 member price: $29.70
# of copies: __________

Portable UNIX
Douglas Topham
1-57926-2 $14.95 member price: $12.71
# of copies: __________

UNIX, Self-Teaching Guide
George Leach
1-57924-6 $19.95 member price: $16.95
# of copies: __________

Object Oriented Programming with Turbo C++
Keith Weiskamp
1-52466-2 $24.95 member price: $21.20
# of copies: __________

Obfuscated C and Other Mysteries
Don Libes
1-57805-3 $39.95 member price: $33.96
# of copies: __________

Finding It On the Internet
Paul Gilster
1-03857-1 $19.95 member price: $16.95
# of copies: __________

Internationalization: Developing Software for Global Markets (pub. date: 1/95)
Tuoc Luong $29.95 member price: $25.45
# of copies: __________

Adventures in UNIX Network Applications Programming
Bill Rieken
1-52858-7 $39.95 member price: $33.96
# of copies: __________

UNIX Shell Programming, 3e
Lowell Jay Arthur
1-59941-7 $29.95 member price: $25.45
# of copies: __________

The UNIX Command Reference Guide
Kaare Christian
1-85580-4 $32.95 member price: $28.01
# of copies: __________

Berkeley UNIX: A Simple & Comprehensive Guide
James Wilson
1-61582-X $40.95 member price: $34.80
# of copies: __________

Payment enclosed, plus sales tax
☐ VISA ☐ Mastercard
☐ American Express
Card No. ____________________________

Name ________________________________
Firm ________________________________
Address _______________________________________
City/State/Zip ____________________________
Signature ______________________________
(order invalid unless signed)
GLOBAL NETWORKS
Computers and International Communication
edited by Linda M. Harasim
Global Networks takes up the host of issues raised by the new networking technology that now links individuals, groups, and organizations in different countries and on different continents. The twenty-one contributions focus on the implementation, application, and impact of computer-mediated communication in a global context.
340 pp. $29.95 hardcover

THE NETWORK NATION
Human Communication via Computer
Revised Edition
Starr Roxanne Hiltz and Murray Turoff
"The Network Nation... contained a fascinating vision. ... It is a place where thoughts are exchanged easily and democratically and intellect affords one more personal power than a pleasing appearance does. Minorities and women compete on equal terms with white males, and the elderly and handicapped are released from the confines of their infirmities to skim the electronic terrain as swiftly as anyone else." — Teresa Carpenter, Village Voice
580 pp. $24.95 paperback

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 do in the future.
279 pp. $24.95 paperback

TECHNOLOGY 2001
The Future of Computing and Communications
edited by Derek Leebaert
Researchers, executives, and strategic planners from inside the companies and laboratories that have shaped today's information age forecast the merging technologies that could well define the computing and communications environment that lies ahead.
392 pp. $14.95 paperback

THE DIGITAL WORD
Text-Based Computing in the Humanities
edited by George P. Landow and Paul Delany
This book explores the larger realm of the knowledge infrastructure where texts are received, reconstructed, and sent over global networks.
Technical Communication and Information Systems series 384 pp. $39.95 hardcover

SOCIOMEDIA
Multimedia, Hypermedia, and the Social Construction of Knowledge
edited by Edward Barrett
SocioMedia continues the assessment of hypertext and hypermedia systems begun in Text, Context, and HyperText and The Society of Text. It examines the use of integrated multimedia to support social or collaborative research, learning, and instruction in the university, one of the best environments for developing and analyzing the effects of computing technologies on our understanding of complex sets of information.
Technical Communications and Information Systems Series 360 pp. $50.00 hardcover

CONNECTIONS
New Ways of Working in the Networked Organization
Lee Sproull and Sara Kiesler
"Sproull and Kiesler raise crucial questions about our technical and particularly our human strategies as a producing society."
— Howard Webber, Sloan Management Review
228 pp. $21.95 paperback

TECHNOBABBLE
John A. Barry
"A serious study of the language of the new technocracy."
288 pp. $12.50 paperback

Payment enclosed Purchase Order Attached Charge to my: Master Card VISA
Card # exp. Signature

Total for book(s) $3.00
Postage for North American addresses
Canadian customers add 7% GST
Total for book(s) & postage

Name
Address
City State Zip Phone FAX

Make checks payable to
THE MIT PRESS
55 Hayward Street, Cambridge, MA 02142-1399 USA
To order by phone, call (617) 625-8569 or (800) 356-0343. E-mail order to mitpress-orders@mit.edu. The operator will need this code: UNIX1.
Prentice Hall PTR is pleased to recommend the following titles to USENIX members...

**USENIX members receive a 15% discount on orders**

<table>
<thead>
<tr>
<th>Title</th>
<th>Authors</th>
<th>ISBN-10</th>
<th>ISBN-13</th>
<th>List Price</th>
<th>Member Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX System Administration Handbook, Second Edition</td>
<td>Evi Nemeth/Garth Snyder</td>
<td>0-13-151051-7</td>
<td>(15105-0)</td>
<td>$48.00</td>
<td>$40.80</td>
</tr>
<tr>
<td>Object-Oriented Modeling and Design</td>
<td>James Rumbaugh</td>
<td>0-13-629841-9</td>
<td>(62984-0)</td>
<td>$54.00</td>
<td>$45.90</td>
</tr>
<tr>
<td>Zen and the Art of the Internet, Third Edition</td>
<td>Brendan Kehoe</td>
<td>0-13-121492-6</td>
<td>(12149-1)</td>
<td>$23.95</td>
<td>$20.36</td>
</tr>
<tr>
<td>The Magic Garden Explained</td>
<td>Bernard Goodheart/James Cox</td>
<td>0-13-098138-9</td>
<td>(09813-7)</td>
<td>$38.00</td>
<td>$32.30</td>
</tr>
<tr>
<td>SCO Performance Tuning Handbook</td>
<td>GinaMiscovich/David Simons</td>
<td>0-13-102690-9</td>
<td>(10269-9)</td>
<td>$42.00</td>
<td>$35.70</td>
</tr>
<tr>
<td>Object-Oriented Programming</td>
<td>Peter Coad/Jill Nicola</td>
<td>0-13-032616-X</td>
<td>(03261-5)</td>
<td>$48.00</td>
<td>$40.80</td>
</tr>
<tr>
<td>Internetworking with TCP/IP, Vol. III Client Server Programming and Applications for the BSD Socket Version</td>
<td>Douglas E. Comer and David L. Stevens</td>
<td>0-13-474222-2</td>
<td>(47422-1)</td>
<td>$53.00</td>
<td>$45.05</td>
</tr>
<tr>
<td>Internetworking with TCP/IP, Vol. III Client Server Programming and Applications for the AT&amp;T TLI Version</td>
<td>Douglas E. Comer and David L. Stevens</td>
<td>0-13-474230-3</td>
<td>(47423-9)</td>
<td>$53.00</td>
<td>$45.05</td>
</tr>
<tr>
<td>The Internet Message: Closing the Book with Electronic Mail</td>
<td>Marshall T. Rose</td>
<td>0-13-092941-7</td>
<td>(09294-0)</td>
<td>$50.00</td>
<td>$42.50</td>
</tr>
<tr>
<td>The Standard C Library</td>
<td>PJ Plauger</td>
<td>0-13-131509-9</td>
<td>(13150-8)</td>
<td>$37.80</td>
<td>$32.13</td>
</tr>
<tr>
<td>All About Administering the NIS+, Second Edition</td>
<td>Rick Ramsey</td>
<td>0-13-309576-2</td>
<td>(30957-5)</td>
<td>$42.00</td>
<td>$35.70</td>
</tr>
<tr>
<td>The Simple Book: An Introduction to Internet Management</td>
<td>Marshall T. Rose</td>
<td>0-13-177254-6</td>
<td>(17725-3)</td>
<td>$55.00</td>
<td>$46.75</td>
</tr>
<tr>
<td>Networking Operations on UNIX SVR4</td>
<td>Mike Padavano</td>
<td>0-13-613555-2</td>
<td>(61355-4)</td>
<td>$50.00</td>
<td>$42.50</td>
</tr>
<tr>
<td>Solaris Porting Guide</td>
<td>Sunsoft ISV Engineering</td>
<td>0-13-030396-8</td>
<td>(03039-5)</td>
<td>$52.00</td>
<td>$44.20</td>
</tr>
<tr>
<td>Multiprocessor System Architectures</td>
<td>Ben Catanzaro</td>
<td>0-13-089137-1</td>
<td>(08913-6)</td>
<td>$44.00</td>
<td>$37.40</td>
</tr>
<tr>
<td>The HP-UX System Administrator's &quot;How To&quot; Book</td>
<td>Marty Poniatowski</td>
<td>0-13-099821-4</td>
<td>(09982-0)</td>
<td>$32.00</td>
<td>$27.20</td>
</tr>
<tr>
<td>UNIX System V Performance Management</td>
<td>Phyllis Eve Bregman and Sally A. Browning</td>
<td>0-13-016429-1</td>
<td>(01642-8)</td>
<td>$29.95</td>
<td>$25.45</td>
</tr>
<tr>
<td>SCO® UNIX® Operating System System Administrator's Guide</td>
<td>Santa Cruz Operation</td>
<td>0-13-012568-7</td>
<td>(01256-7)</td>
<td>$39.00</td>
<td>$33.15</td>
</tr>
</tbody>
</table>

**HERE'S HOW TO ORDER:**

**CALL**

800-880-6818

**OR WRITE:**

CompuBooks
Route 1, Box 271-D,
Cedar Creek, TX 78612

**OR INTERNET:**

70007.1333@CompuServe.com
(GO CBK on CompuServe)

**WE SHIP ANYWHERE!**

**FOR MORE INFORMATION, OR QUANTITY ORDERS, PLEASE CALL 201-592-2657**
A UNIQUE OFFER
ON THE BEST IN UNIX
FOR USENIX MEMBERS

□ THE INTERNET GUIDE FOR NEW USERS
D. Dern
hardcover, 016510-6, $40.00,
MEMBER PRICE $32.00
paperback, 016511-4, $27.95,
MEMBER PRICE $22.36

□ INTERNET FOR EVERYONE
R. Wiggins
hardcover, 067018-8, $29.95,
MEMBER PRICE $23.96
paperback, 067019-6, $45.00,
MEMBER PRICE $36.00

□ THE ESSENTIAL INTERNET INFORMATION GUIDE
J. Manger
707905-1, paperback, $27.95,
MEMBER PRICE $22.36

□ THE INFORMATION BROKERS HANDBOOK, Second Edition
S. Rugge
911878-x, paperback, $34.95,
MEMBER PRICE $27.96
Available December 1994

□ SAA AND UNIX: IBM's Open System Strategy
M. Killen
034607-0, $40.00,
MEMBER PRICE $32.00

□ A STUDENT'S GUIDE TO UNIX
H. Hahn
025511-3, paperback, $28.00,
MEMBER PRICE $22.40

□ UNIX DEVELOPER’S TOOL KIT
K. Leininger
911836-4, $65.00,
MEMBER PRICE $52.00

□ UNIX SECURITY: A Practical Tutorial
N. Arnold
002560-6, $24.95,
MEMBER PRICE $19.96

□ THE UNIX AUDIT: Using UNIX to Audit UNIX
M. Grottola
025127-4, $32.95,
MEMBER PRICE $26.36
Available November 1994

□ UNIX: A Database Approach
S. Das
015745-6, $29.95,
MEMBER PRICE $23.96
Available November 1994

I am a member of USENIX Association. Please send me the books I have indicated at the member special rate. I have added $3.00 postage and handling for the first book ordered, $1.00 for each additional book, plus my local sales tax.

Check or money order is enclosed—payable to McGraw-Hill, Inc.
Charge my □ Visa □ Mastercard □ Discover □ Amex
Account # ________________________________
Expiration Date ________________________________
Signature ________________________________

USENIX Membership # ________________________________

Bill & Ship To:
Name ________________________________
Street ________________________________
City, State, Zip ________________________________
Daytime Phone # ________________________________

Send or Fax Orders to:
McGraw-Hill, Inc.
Attn: Rosa Perez
11 West 19th Street—4th Floor
New York, New York 10011
Fax: 212-337-4092

If I am not completely satisfied, I will return the book(s) within 15 days for a full refund or credit. Satisfaction unconditionally guaranteed. Prices subject to change without notice. We can only accept orders within the continental USA.

72 ;login: VOL. 21 - NO. 4

Today, these are the reasons PLATINUM Solutions leads the drive to connect companies and their customers. Tomorrow, our mastery of the Internet, our commitment to customer education, our ability to create distributed systems, and our development of systems software will define our opportunities. And yours.

Our Locus Services Division is a national leader in UNIX development for open systems. Opportunities currently exist in Southern California (Los Angeles and San Diego) and Burlington, Massachusetts for Systems Software Developers. We are seeking individuals with experience in one of the following areas:

**Systems Software Developers**

- UNIX Kernel Development (e.g. Virtual Memory, Virtual File Systems, Networking)
- Device driver development - UNIX Communications (e.g. terminals, Ethernet) or Windows NT/95
- MPP performance measurement
- UNIX operating systems porting
- Xserver and Xtoolkit internals knowledge with X11 applications programming
- Motif troubleshooting support with source code experience a plus

Set a new standard in Systems Software Development. The PLATINUM Standard. We offer an excellent integrated salary and benefits package, flexible hours, and a casual work atmosphere. Send your resume to: PLATINUM Solutions, Inc., Attn: Vince Melillo, 9800 La Cienega Blvd., Inglewood, CA 90301-4440. Fax: (310) 337-5292. email: vincem@platsol.com. Visit our WWW site at www.platsol.com. We are an Equal Opportunity Employer and actively encourage women, minorities and international candidates to apply.
Contract Consulting Opportunities

in California for
Unix Systems Administrators

Systems Partners is an information technology consulting company serving Fortune 100 companies in the San Francisco Bay Area, Sacramento, and the Silicon Valley.

Why Would You Prefer to Work with Systems Partners?

• We treat consultants as business partners and share information about bill rates, margins, roles and deliverables.

• We provide ongoing Account Management to proactively support consultants throughout an engagement.

• We implement a proven Quality Program uniquely designed to clarify deliverables and guarantee project success.

• We offer a consultant benefits package that includes a 401K plan, direct deposit, health & dental insurance, long-term disability and pre-tax expense program.

• We are committed to continual learning through our Technical University, Seminar Series, and Special Interest Groups.

• We have a history of success with over 70 Fortune 100 companies around the Bay Area including preferred vendor services to VISA, Hewlett Packard, Charles Schwab, Wells Fargo, Amdahl and others.

Unix Systems Administrators at all levels are in high demand for both contract and contract-to-hire opportunities.

Platforms, tools and skills in demand include HP-UX, AIX, Sun Solaris, Perl, Shell Scripting, CGI, TCP/IP, NFS, NIS, Firewalls, and C Programming.

We Work for You to provide top-notch, challenging contract consulting opportunities that allow you to build your skills and stay competitive.

Browse Our Web Site for more information about our open engagements, consultant benefits package, user groups, special events and training programs:

http://www.syspart.com

For an update on our current consulting opportunities, call our Resource Managers, Steven Laine and Alex Schwab at 1-800-479-4116.

Systems Partners

2 Theatre Square, Suite 315, Orinda, CA 94121
Tel: 1-800-479-4116, Fax: 1-510-253-2576, E-mail: info@syspart.com
Bluestone is an industry leader in providing Product Development, Training, and Technology Transfer solutions to Fortune 500 Corp's in the Advanced Development Marketplace. Our progressive environment (learn, apply, prosper) as a technology driven firm enables us to establish long-term relationships with many of the world's leading firms. It is our directive to attract and retain the highest caliber professionals the industry has to offer. As Bluestone continues to mature it’s national and international presence, we seek individuals who not only understand today's technology, but share in the commitment to grow and learn with us. The Bluestone model allocates 15% of each individual’s time to research and learning to ensure that your skills will always coincide with the latest technology in the Client-Server marketplace.

System/Network

Administration positions available locally and worldwide.

JOIN BLUESTONE'S ELITE GROUP OF PROFESSIONALS
CHALLENGE • DIVERSIFICATION • EXCELLENCE

Contact Donna DeCarolis

Bluestone
Transferring Advanced Development Technology

1000 Briggs Road,
Mt. Laurel, NJ 08054
Phone: (609) 727-4600 x1010
Fax: (609) 727-1318
email: donnad@bluestone.com
The 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:. At least one member of the group must be a current member of the Association. Send additions and corrections to: <login@usenix.org>.

**California**

**Fresno:** The Central California UNIX Users Group has a WWW contact page to which members may post questions or information. For connection information:

- Steve Mitchell
  209 278 5675
  <http://warpig.catl.csufresno.edu/ccuug/ccuug.html>

**Orange County:** Meets the 2nd Monday of each month

- UNIX Users Association of Southern California (UUASC)
  Dave Close
  714 434 7359
  <dave@uuasc.org>

**Colorado**

**Boulder:** Meets monthly at different sites; for membership information and meeting schedule, send email to <fruug-info@fruug.org>.

- Front Range UNIX Users Group
  Lone Eagle Systems Inc.
  636 Arapahoe #10
  Boulder, CO 80302
  Steve Gaede
  303 444 9114
  <gaede@fruug.org>
  <http://www.fruug.org/~fruug>

**Washington, D.C.**

Meets 2nd Tuesday of each month.

- Washington Area UNIX Users Group
  10440 Shaker Drive, Suite 103
  Columbia, MD 21046
  Alan Fedder
  301 621 5500
  <afeedder@mcs.org>

**Florida**

**Orlando:** Meets the 3rd Thursday of each month.

- Central Florida UNIX Users Group
  Bob Boarman
  <bboardman@national.aaa.com>

**Western:** Meets 1st Thursday of each month.

- Florida West Coast UNIX Users Group
  Mike Delucia
  813 882 0770
  <pfl@cfnet.com>

**Georgia**

**Atlanta:** Meets on the 1st Monday of each month in White Hall, Emory University.

- Atlanta UNIX Users Group
  P.O. Box 12241
  Atlanta, GA 30355-2241
  Mark Landry 404 365 8108

**Kansas and Missouri**

Meets on 2nd Tuesday of each month.

- Kansas City UNIX Users Group (KCUG)
  P.O. Box 412622
  Kansas City, MO 64141
  816 891 1093
  <richj@northcs.cps.com>

**Michigan**

**Detroit/Ann Arbor:** Meets on the 2nd Thursday of each month in Ann Arbor.

- Southeastern Michigan Sun Local Users Group and Nameless UNIX Users Group
  Steve Simmons
  office: 313 769 4086
  home: 313 426 8981
  <sce@lokkur.dexter.mi.us>

**New England**

Northern New England UNIX Users Group (NNEUUG)
Meets monthly at different sites.

- Peter Schmitt
  508 289 2877
  Woods Hole Oceanographic Institute
  Woods Hole, MA
  <pschmitt@whoi.edu>

**New Mexico**

**Albuquerque:** ASIGUNIX
<asigunix@rt66.com> meets every 3rd Wednesday of each month.

- Phil Hortz
  505 275 0466
  <prh@bossnet.com>

**New York**

**New York City:** Meets every other month in Manhattan.

- Unigroup of New York City
  G.P.O. Box 1931
  New York, NY 10116
  <uniboard@unigroup.org>
  J. P. Radley
  212 877 0440

**Oklahoma**

**Tulsa:** Meets 2nd Wednesday of each month.

- Tulsa UNIX Users Group,
  SUSR Bill Hunt
  918 494 4848
  <bhunt@tulsix.utulsa.edu>
  Mark Lawrence
  918 749 7498
  <lawrence@tulsix.utulsa.edu>

**Texas**

**Austin:** Meets 3rd Thursday of each month.

- Capital Area Central Texas UNIX Society (CACTUS)
  P.O. Box 9786
  Austin, TX 78766-9786
  Ronald S. Woan
  512 838 1254
  <president@cactus.org>
  <http://cactus.org>

**Dallas/Fort Worth:** Meets the 1st Thursday of each month.

- Dallas/Fort Worth UNIX Users Group
  P.O. Box 867405
  Plano, TX 75086
  Evan Brown
  214 519 3577
Texas (cont)

Houston: Meets 3rd Tuesday of each month.
• Houston UNIX Users Group (Hounix) answering machine: 713 684 6590
  Jack Gilbert, President 713 862 3637 
  <jack@hounix.org>

Washington

Seattle: Meets monthly.
• Seattle UNIX Group
  Bill Campbell 206 947 5591
  P.O. Box 820
  Mercer Island, WA 98040-0820
  <slug@seaslug.org>
  <http://www.seaslug.org>

Canada

Calgary: Meets 4th Tuesday of each month.
• Calgary UNIX Users Group (CUUG)
  David Marwood
  <postmaster@cuug.ab.ca>
  <http://www.cuug.ab.ca:8001>

Manitoba: Meets 2nd Tuesday of each month.
• Manitoba UNIX User Group (MUUG)
  Bary Finch, President
  204 934 1690
  <info@muug.mb.ca>

Ottawa: Meets 3rd Wednesday of each month except July and August
• The Ottawa Carleton UNIX Users Group (OCUUG)
  Dave Blackwood
  <dave@revcan.ca>

Toronto:
• 143 Baronwood Court
  Brampton, Ontario
  Canada L6V 3H8
  Evan Leibovitch
  416 452 0504
  <evan@telly.on.ca>

Quebec: Meets first Wednesday every 3rd month.
• Administrateurs de Système UNIX du Quebec (ASUQ)
  Université de Montreal, Dept. IRO
  C.P. 6128, Succ. Centre-Ville
  Montreal, Quebec, Canada, H3C 3J7
  514 343 7480

System Administration Groups

Back Bay LISA (BBLISA)

New England forum covering all aspects of system and network administration, for large and small installations. Meets monthly, at MIT in Cambridge, MA.

• For information, contact:
  • J. R. Oldroyd 617 227 5635
    <jr@opal.com>
  • Mailing list subscription:
    <bblisa-request@bblisa.org>
  • Mailing list postings:
    <bblisa@bblisa.org>
  • For current calendar of events:
    finger <bblisa@finger.bblisa.org>

Bay LISA (California)

Meets 3rd Thursday of each month at Synopsys in Mountain View, CA. For more information, please contact:
<bblisa@baylisa.org>
or
<http://www.baylisa.org>

TCSA (D.C.)

The Twin Cities System Administrators Group meets on the 3rd Thursday of each month in the Twin Cities area of Minnesota.

• Users can be a friend of the system administrator, but they will never be able to be a peer.” We’re here to meet, interact, support, leverage, and otherwise make your vocation a more fruitful one. For more information, send “info dc-sage” to:
  <info@tcsa.org>
  or
  <majordomo@mrj.com>
  Carolyn J. Sienkiewicz
  <cjs@chokey.mo.md.us>
  Brad Knowles
  <bkowles@aol.net>

$GROUPNAME (New Jersey)

$GROUPNAME is an organization in New Jersey formed to facilitate information exchange pertaining to the field of UNIX system administration. For more information, send “infogroupname” to <majordomo@plts.org>.

Tom Limoncelli <tal@big.att.com>

New York Systems Administrators (NYSA)

Meets 2nd Monday of each month.
<nysa-request@esm.com>
914 472 3635 or 472 3635

North Carolina System Administrators Group

The North Carolina System Administrators Group meets on the 2nd Monday each month around the Research Triangle Park area.
• Amy Kreiling 919 962 1843
  <kreiling@cs.unc.edu>
• William E. Howell 919 941 4868
  <william.howell@glaxo.com>

TCSA (D.C.)

TCSA meets on the 3rd Thursday of each month in the Twin Cities area of Minnesota.
<http://www.tcsa.org>
<info@tcsa.org>
## Calendar of Events

This is a combined calendar of conferences, symposia, and standards meetings. If you have an event that you wish to publicize, please contact <login@usenix.org>. For complete USENIX conference and symposia listings see URL [http://www.usenix.org/events/general.html](http://www.usenix.org/events/general.html).

For an up-to-date, comprehensive, and easy-to-access information resource on the Internet, covering events all over the world, consult the WWW Virtual Library on Conferences at Fraunhofer-IAO. [http://www.iao.fhg.de/Library/conferences](http://www.iao.fhg.de/Library/conferences)

* = events sponsored by the USENIX Association.

### 1996

#### August

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 - 9</td>
<td>SIGGRAPH, New Orleans, LA</td>
<td></td>
</tr>
<tr>
<td>5 - 9</td>
<td>Interex ’96, San Diego, CA</td>
<td></td>
</tr>
<tr>
<td>28 - 30</td>
<td>SIGCOMM, Palo Alto, CA</td>
<td></td>
</tr>
</tbody>
</table>

#### September

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 - 5</td>
<td>GUUG, Leipzig, Germany</td>
<td></td>
</tr>
<tr>
<td>9 - 11</td>
<td>ACM SIGOPS European Workshop, Ireland</td>
<td></td>
</tr>
<tr>
<td>16 - 20</td>
<td>NetWorld+Interop ’96, Atlanta, GA</td>
<td></td>
</tr>
<tr>
<td>18 - 20</td>
<td>AUUG, Melbourne, Australia</td>
<td></td>
</tr>
<tr>
<td>30 -</td>
<td>Oct 4 * LISA ’96, Chicago, IL</td>
<td></td>
</tr>
</tbody>
</table>

#### October

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 4</td>
<td>ASPLOS VII, Cambridge, MA</td>
<td></td>
</tr>
<tr>
<td>7 - 11</td>
<td>NetWorld+Interop ’96, Paris, France</td>
<td></td>
</tr>
<tr>
<td>8 - 10</td>
<td>UNIX Expo, New York City</td>
<td></td>
</tr>
<tr>
<td>10 - 16</td>
<td>OOPSLA ’96, San Jose, CA</td>
<td></td>
</tr>
<tr>
<td>23 - 25</td>
<td>IEEE Symposium on Reliable Distributed Systems, Niagara, Canada</td>
<td></td>
</tr>
<tr>
<td>27 - 28</td>
<td>*IWOOOS ’96, Seattle, WA</td>
<td></td>
</tr>
<tr>
<td>28 - 31</td>
<td>*OSDI II, Seattle, WA</td>
<td></td>
</tr>
<tr>
<td>30 -</td>
<td>Nov 2 ROSE ’96, Bucharest, Romania</td>
<td></td>
</tr>
</tbody>
</table>

#### November

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 - 8</td>
<td>Open Systems World/ FedUNIX, Washington, DC</td>
<td></td>
</tr>
<tr>
<td>4 - 8</td>
<td>UNIX Network Security, Washington, DC</td>
<td></td>
</tr>
<tr>
<td>9 - 14</td>
<td>DECUS, Anaheim, CA</td>
<td></td>
</tr>
<tr>
<td>17 - 22</td>
<td>ACM IEEE-CS Supercomputing ’96, Pittsburgh, PA</td>
<td></td>
</tr>
<tr>
<td>18 - 20</td>
<td>Electronic Commerce, Oakland, CA</td>
<td></td>
</tr>
<tr>
<td>25 - 29</td>
<td>NetWorld+Interop ’96, Sydney, Australia</td>
<td></td>
</tr>
</tbody>
</table>

#### December

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 - 13</td>
<td>IETF, San Jose, CA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JUS UNIX Fair</td>
<td></td>
</tr>
</tbody>
</table>

### 1997

#### January

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 - 10</td>
<td>USENIX, Anaheim, CA</td>
<td></td>
</tr>
<tr>
<td>6 - 10</td>
<td>USELINUX Conference, Anaheim,CA</td>
<td></td>
</tr>
<tr>
<td>20 - 24</td>
<td>POPL ’97</td>
<td></td>
</tr>
</tbody>
</table>

#### March

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 5</td>
<td>ACM ’97, San Jose, CA</td>
<td></td>
</tr>
<tr>
<td>10 - 14</td>
<td>UniForum, San Francisco, CA</td>
<td></td>
</tr>
</tbody>
</table>

#### April

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 - 11</td>
<td>IETF, Memphis, TN</td>
<td></td>
</tr>
<tr>
<td>7 - 11</td>
<td>Networld+Interop ’97, Singapore</td>
<td></td>
</tr>
<tr>
<td>21 - 26</td>
<td>SANS, Baltimore, MD</td>
<td></td>
</tr>
</tbody>
</table>

#### May

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 - 7</td>
<td>HotOS-VI</td>
<td></td>
</tr>
</tbody>
</table>

#### June

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 - 19</td>
<td>*COOTS, Portland, OR</td>
<td></td>
</tr>
<tr>
<td>16 - 20</td>
<td>SIGPLAN ’97</td>
<td></td>
</tr>
</tbody>
</table>

#### July

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Annual Tc/TK Workshop</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### October

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 - 17</td>
<td>OOPSLA ’97</td>
<td></td>
</tr>
<tr>
<td>27 - 31</td>
<td>*LISA ’97, San Diego, CA</td>
<td></td>
</tr>
</tbody>
</table>

#### 1998

#### January

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>19 - 23</td>
<td>POPL ’98</td>
<td></td>
</tr>
<tr>
<td>26 - 29</td>
<td>7th USENIX Security Symposium</td>
<td></td>
</tr>
</tbody>
</table>

#### June

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 - 19</td>
<td>USENIX, New Orleans, LA</td>
<td></td>
</tr>
</tbody>
</table>

#### December

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 - 11</td>
<td>LISA ’98, Boston, MA</td>
<td></td>
</tr>
</tbody>
</table>

---

*ACM: Association for Computing Machinery*

*ASPL0S: Architectural Support for Programming Languages and Operating Systems*

*AUUG: Australian UNIX Systems Users Group*

*COOTS: Conference on Object-Oriented Technologies and Systems*

*DECUS: Digital Equipment Computer Users Society*

*EurOpen: European Forum for Open Systems*

*FIRST: Forum of Incident Response and Security Teams*

*GURU: Romanian UNIX User Group*

*GUUG: German UNIX Users Group*

*HotOS: Hot Topics in Operating Systems*

*IIEEE: Institute of Electrical and Electronics Engineers*

*IETF: Internet Engineering Task Force*

*INET: Annual Conference of Internet Society*

*IWOOOS: International Workshop on Object-orientation in Operating Systems*

*JUS: Japan UNIX Society*

*LISA: USENIX/SAGE Systems Administration Conference*

*OOPSLA: Object-oriented Programming Systems, Languages and Applications*

*OSDI: Symposium on Operating Systems Design & Implementation*

*POPL: Principles of Programming Languages*

*ROSE: Open Systems in Romania*

*SANS: System Administration, Networking & Security*

*SDNE: Services in Distributed and Networked Environments*

*SIGOPS: ACM Special Interest Group on Operating Systems*

*SIGPLAN: ACM Special Interest Group on Programming Languages*

*SIGSOFT: ACM Special Interest Group on Software Engineering*

*SOSP: ACM Symposium on Operating Systems Principles*

*SUG: Sun User Group*

*UKUUG: United Kingdom UNIX Systems Users Group*

*UniForum: International Association of UNIX and Open Systems Professionals*

*WITI: International Network of Women in Technology*
USENIX

1996-97 Conferences, Symposia, and Workshops for Advanced Computing Systems Professionals and Systems and Network Administrators

If these topics are important to you:

- Systems Administration
- Operating Systems Design and Implementation
- Electronic Commerce
- Linux
- Embedded Systems

then save these dates!

Plan to attend these USENIX events:

- 10th Systems Administration Conference (LISA '96). September 29–October 4, 1996. Chicago, IL
- Embedded Systems. Dates and location to be announced.

For more information:
USENIX, 22672 Lambert St., Suite 613, Lake Forest, CA 92630
Phone: 714.588.8649
Fax: 714.588.9706
Email: conference@usenix.org
WWW: http://www.usenix.org

USENIX, the UNIX and Advanced Computing Systems Technical and Professional Association, offers technical conferences for and by technical professionals.
SAGE, the System Administrators Guild, a special technical group within USENIX, is dedicated to the advancement and recognition of system administration as a profession.