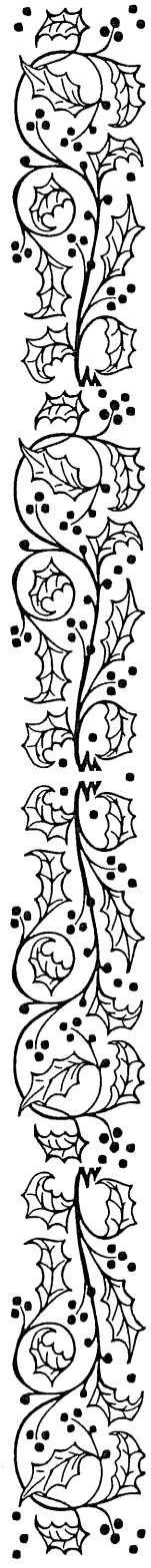
AUUGN

The Journal of AUUG Inc.

Volume 16, Number 6
December 1995



Compliments of the Season to all AUUG members!

Attention to detail!

A UNIX word processor

... plus reviews, Chapter news, and more

AUUG Inc.

UNIX & OPEN SYSTEMS USERS

ISSN 1035-7521

Print post approved by Australia Post - PP2391500002

AUUGN
Volume 16, Number 6
December, 1995

AUUG Membership & General Correspondence

The AUUG Secretary
PO Box 366
Kensington NSW 2033

Tel: (02) 361 5994
Fax: (02) 332 4066
Freephone: 1-800-625-655
E-mail: *auug@auug.org.au*

AUUG Executive

President: Michael Paddon
mwp@aba.net.au
Kodak
173 Elizabeth St.
Coburg VIC 3058

Vice President: Glenn Huxtable
glenn@fs.com.au
Functional Software
PO Box 192
Leederville WA 6903

Secretary: Peter Wishart
pjw@auug.org.au
EASAMS/GEC Marconi Systems
PO Box 4806
Unit 7, 10 Kennedy St.
Kingston ACT 2604

Treasurer: Stephen Boucher
stephen@mtiame.mtia.oz.au
MTIA
509 St. Kilda Road
Melbourne VIC 3004

Committee Members:

Phil McCrea (Past President:)
pmc@syd.dit.csiro.au
Division of Information Technology
CSIRO
Building E6B
Macquarie University NSW 2113

Frank Crawford
frank@atom.ansto.gov.au
ANSTO
Private Mail Bag 1
Menai NSW 2234

Lucy Chubb
lucyc@sw.oz.au
Softway Pty. Ltd.
PO Box 305
Strawberry Hills NSW 2021

Chris Maltby
chris@sw.oz.au
Softway Pty. Ltd.
PO Box 305
Strawberry Hills NSW 2021

David Purdue
David.Purdue@aus.sun.com
SunSoft
119 Willoughby Rd.
Crows Nest NSW 2065

AUUGN Business Manager

Elizabeth Egan
auug@auug.org.au
PO Box 366
Kensington NSW 2033

Table of Contents

Contribution deadlines for AUUGN in 1995/96

Vol 17, #1 (February '96): February 14th

Vol 17, #2 (April '96): March 22

Vol 17, #3 (June '96): May 24

Vol 17, #4 (August '96): July 19

Vol 17,#5 (October '96): September 20

Vol 17, #2 (December '96): November 22

AUUGN Correspondence

Please send all correspondence regarding AUUGN to:

AUUGN Editor
PO Box 366
Kensington NSW 2033

E-mail: *auugn@auug.org.au*

Submission Guidelines

Submission guidelines for AUUGN contributions are regularly posted on the aus.org.auug news group. They are also available from the AUUG World Wide Web site at http://www.auug.org.au

Alternately, e-mail to the above correspondence address, requesting a copy.

AUUGN Back-Issues

A variety of back-issues of AUUGN are still available; for price and availability details, please contact the AUUG Secretariat, or write to:

AUUG Inc.
Back Issues Department
PO Box 366
Kensington NSW 2033
Australia

Conference Proceedings

A limited number of copies of the Conference Proceedings from previous AUUG Conferences are still available, at $50 each for members, and $60 for non-members. Contact the AUUG Secretariat for details.

Mailing Lists

Inquiries regarding the purchase of the AUUGN mailing list should be directed to the AUUG Secretariat. Telephone (02) 361 5994 during business hours, or Fax (02) 332 4066.

Editorial

Phil Anderson <phila@osa.com.au>

It's the close of another year once more; regrettably, due to the usual range of personal and other crises (ahhh, ain't Christmas grand) you'll be seeing this issue well after you've been through all the festive guff and are happily back at your desks, fondling your workstations (*hmmm...*).

There are a some folk I'd like to thank for their efforts and support during 1995, most notably Frank & Jadoga Crawford for a whole slew of contributions and subediting; Phil McCrea for his prodigious output and encouragement; Janet Jackson for tricks, traps and keeping an eye on the West; Catrina Dwyer and Liz Egan for their coordination skills; and Michael Paddon for taking on the grave responsibility of the dreaded President's Report! There are also plenty of others who deserve credit for their help; thank you one and all!

❧

Oh yeah; I did get to the polo a week later, and the scariest thing that happened was that I actually started to enjoy the game, and forgot all about that performance art angle. Damn! It was pretty exciting at times... next time the Spring Carnival rolls around (or your local equivalent) see if you can dig up a match or two, and take a picnic and some pals; have some fun and cast your IT cares aside...

❧

While we're on the subject of Open Systems philosophies, I'd like to invite those of you who have Web browsers to take a look at the current efforts of my new chums, Open Software Associates, in bringing client-server applications to the Web. Their address is http://www.osa.com.au❖

President's Report

Michael Paddon <mpaddon@aba.net.au>

Crystal ball gazing is a perilous activity at the best of times, but I always find myself dabbling in the business of second guessing the future around the new year. Indeed, if you are like me, your business and career hinge upon correctly predicting at least a few key directions.

Of course, if we could select our strategies on rational technical criteria it would all be easy. But there are dark "market forces" at work that confuse the issue and tend to litter history with good technologies that failed to gain "acceptance". Anyone remember the NeWS windowing system? Instead we are stuck with a marvel of engineering over design: X11. How did the System V IPC and file locking ever become standard, when there were alternatives that actually worked well?

Enough reminiscing. What are my predictions for 1996?

This is definitely the year of Java. James Gosling, one of the key architects of the new language, has been an advocate of high level, real time downloadable languages for some time (the abovementioned NeWS system being an example of his earlier work). This time around, however, the applet philosophy is using the WWW as a springboard; and the pent up demand for highly responsive web content is creating an environment extremely conducive to Java's success.

Apart from providing a quantum leap in the functionality of the web for the average user, Java will make programming a skill of equal importance to design and layout. This is welcome news for professional software people. Even nicer is the fact that Java is a real object oriented language (based on C++) in which it is feasible to build industrial strength solutions.

This is also the year of a strong resurgence of UNIX as the key platform for delivering robust Internet services. Lots of businesses have made "strategic" decisions for also-rans like NT, but without understanding that the fastest, most functional and most secure server software is usually available only under UNIX. Given the fundamental health and strength of the free software community it is only likely that the gap between UNIX and its competition in the server world will widen.

The other force that will be driving UNIX forward into the future is the availability of high quality, free distributions. I've already waxed lyrical about NetBSD in a previous column, but FreeBSD and Linux are also extremely viable systems

We will begin to see increasing interest, and perhaps even some implementations, of IPv6 (the next version of the Internet protocols). Free UNIX systems are in a unique position to be in the first wave of machines that support the new functionality due to the speed of their release cycles and the likely interest of their development teams.

What won't happen this year? Standards wars are a dead issue. If vendors don't implement POSIX they may as well not bother knocking on my door. Superstandards (the drawing together of POSIX, X/Open, etc) are not very interesting.

With any luck the C++ standards body will stop making massive changes. Last year's first prize went to the change of scope for variables declared in a for loop... instantly breaking millions of lines of code.

Back to the here and now. There have been a few developments of note over the past few months, although Christmas tends to be a quiet time for AUUG.

It is summer (well, everywhere except Melbourne), and it looks like there will be a summer conference in every capital city this year. Try to get along to the one nearest you (details are available on http://auug.org.au/, or give us a call if you don't have web access). These events are small, focussed and are a great chance to meet other local exports and enthusiasts.

It is with pleasure that I announce appointment of Enno Davids as Conference Chair and Adrian Booth as Programme Chair of the AUUG 96 Winter Conference.

It may come as a surprise to you, but work is already well under way for the coming September, and Enno, Adrian and a hidden cast of thousands are already putting in a significant effort. You will be seeing a call for papers soon, and I encourage you to think about speaking or recommending the conference to a colleague who may be interested in presenting. Or you may want to volunteer some time... we have lots of interesting jobs.

Finally, as some may know, Brenda Parsons has resigned her position on the NSW chapter committee. I'd like to take this opportunity, on behalf of AUUG, to thank Brenda for all that she has done for us over the years, and in particular for the energy she devoted to getting the NSW chapter running.❖

Conferences & Announcements

The USENIX Association presents:

LIFETIME ACHIEVEMENT AWARD

to

The SOFTWARE TOOLS USERS GROUP

Principal Recipients and Keepers of the Flame

Dennis Hall
Deborah Scherrer
Joseph Sventek

Originators and Key Inspiration

Brian Kernighan
P. J. Plauger

Major Contributors, named in the award

Before the general availability of UNIX, the Software Tools project popularized a new vision of operating system software, offering a bridge to portability and power for those limited by proprietary operating systems.

With its extraordinary focus on building clean, portable, reusable code shared amongst multiple applications and runable on virtually any system, the Software Tools movement established the tradition of empowering users to define, develop, control, and freely distribute their computing environment. The contributions of STUG in retrospect can be seen to have been vital.

The USENIX Lifetime Achievement Award recognizes and celebrates singular contributions to the UNIX community in both intellectual achievement and unparalleled service. Past recipients of the USENIX Award are the Computer Science Research Group at the University of California at Berkeley for producing the UNIX BSD releases, Van Jacobson and Mike Lesk for their contributions to networking technology, and Tom Truscott, Steve Bellovin and Jim Ellis for their work in creating USENET.

Simultaneously, The USENIX Association, acting on behalf of The Software Tools Users Group presents:

The SOFTWARE TOOLS USERS GROUP AWARD

to:

Michael Tiemann

Michael Tiemann's work in C++ led to fundamental contributions to the GCC, the GNU C Compiler, which has had an unparalleled influence upon the availability of efficient and standard code

on a vast number of hardware platforms. GCC has provided a development base for thousands of projects.

The Software Tools User Group Award recognizes significant contributions to the general community which reflect the spirit and character demonstrated by those who came together in the Software Tools User Group (STUG). Therefore, recipients of the Software Tools Award exhibit one or both of these traits in a conspicuous manner: a contribution to the reusable code-base available to all, or the provision directly to users in a widely-available form of a significant, enabling technology.

Steve Johnson, President of the USENIX Association, will be announcing both 1996 awards on Wednesday, January 24, at the Annual USENIX Technical Conference, January 22-26, 1996, in San Diego, California.

❧

USENIX is the UNIX and Advanced Computing Systems Technical and Professional Association. Since 1975 the USENIX Association has brought together the community of engineers, scientists, and technicians working on the cutting edge of the computing world. The USENIX conferences have become the essential meeting grounds for the presentation and discussion of the most advanced information on the developments of all aspects of computing systems.

For more information about USENIX events, please: - Contact USENIX Conference Office, 22672 Lambert Street, Suite 613, Lake Forest, CA USA 92630, Telephone +1 714 588 8649, Fax +1 714 588 9706

Email: **conference@usenix.org**

Access the USENIX Resource Center on the World Wide Web--the URL is **http://www.usenix.org**

Email to our mailserver at **info@usenix.org**. In your message include the line "send conferences catalog."❖

Review:

A Free Word Processor for UNIX

Michael Paddon <mwp@aba.net.au>

It all came to a head when I refused to sully gunbuster's disk with a copy of Windows. Gunbuster, you see, is my new toy: a PC that almost looks like a workstation, running a slightly hacked copy of NetBSD 1.1. Sure, I run DOS on it so I can boot Doom, but I had no intention of compounding that crime (or complicating my life) by running the W program.

Simple enough, except that Linda saw it differently. "I want to run a word processor... something where I can pull down the functions and see what I'm doing", she insisted. This whole concept of using the computer as a tool to do a job seemed quite foreign to me, but my wife seemed to know exactly what she wanted. And who am I to argue?

I was adamant, however, that if word process we must, we were jolly well going to do it under UNIX. So began the great search for the right tool (it had to do everything) at the right price (for free seemed reasonable). In the meantime I pointed Linda at xcoral as a stop gap, which she didn't like; and at vi which she did like after she realised she knew it anyway since she had been using it to compose email for months.

An extended search of the Web threw up two interesting packages that looked like they might fill Linda's shopping list. The first was a program called Xword, produced by the Hungry Programmers. These are the same poorly fed people who are responsible for Lesstif, the excellent free Motif clone.

Xword is eventually intended to be a fully featured X11 WYSIWYG word processor, based on top of an SGML engine. Eventually. Unfortunately, Xword is still in pre-alpha development and the current snapshots are not yet functional enough to be useful (as is pointed out by the authors). Xword will definitely be worth a look in 6 to 12 months, but is currently only of interest if you want to pitch in and help develop it.

The second package I turned up looked far more promising. LyX is a "near WYSIWYG" front end for the LaTeX typesetting system, an idea I found intriguing

AUUG 96

18 - 20 September
World Congress Centre, Melbourne, Australia

AUUG Inc
UNIX AND OPEN SYSTEMS USERS

CALL FOR PAPERS

Anyone involved in AUUG, either as a member or just as an observer would be aware that the next winter conference will be held at the World Congress Centre in Melbourne, between 18th and 20th September. As usual this will be preceded by a number of tutorials, covering two days, the 16th and 17th.

As this event is the biggest open systems event in Australia (attended by over 800 at AUUG95!) we are currently seeking papers and tutorials for this exciting event.

The program committee is willing to consider proposals falling into the following areas:

- Technical aspects of Unix and Open Systems
- Networking, Internet and World Wide Web
- Commercial Experience and Case Studies

Presentations may either be constructed as tutorials, technical papers, or management studies. Technical papers are designed for those who require in-depth practical knowledge, whereas management studies will deal with business applications of these technologies. Presentations should be designed for a duration of 30 minutes, with additional time allowed for questions.

The tutorials, which may be of either technical or management orientation, providing a more thorough presentation, running either a half or a full day, and giving those attending useful knowledge that they can use in their current field.

While many people attend in order to obtain new information, knowledge is only valuable if it is shared. Without such sharing, it is impossible to verify that it is correct or even useful. By presenting either an outline or details of current projects or research interests it is possible to find others who are either working in the same field and can assist you, or who are interested in your results and willing to support you.

So, if you have anything that you feel is valuable to the computing community within Australia, whether it is in a technical area or a management area, you should consider presenting at AUUG96. It will certainly be aired before one of the biggest and most diverse cross-section of local computer people and is very likely to attract interest.

Rather than sitting back, you should submit an abstract to the program committee, outlining you presentation and see if you can join those others who are trying to influence the direction of computing in Australia.

SUBMISSION GUIDELINES

Those proposing to present papers should submit an extended abstract (1-3 pages) and a brief biography.

Those submitting tutorial proposals should submit a plan for the tutorial and a brief biography.

Please indicate on your proposal whether it is a paper or tutorial, and for tutorials whether they are half day or full day.

Important Dates

Abstracts/Proposals due
15 May 1996

Authors notified
5 June 1996

Final copy due
15 August 1996

Conference
18-20 September 1996

Proposals should be sent to:

AUUG Inc
PO Box 366
KENSINGTON NSW 2033

Email: auug96@auug.org.au

and attractive given that LaTeX is my document production system of choice.

What does "near WYSIWYG" mean? Rather than attempt the impossible task of showing what the document will look like on paper, LyX shows (and allows you to edit) a view of the document that is tuned for CRT display. For instance, paragraphs or lists or font sizes may come out differently in hard copy, but you can see them represented clearly on the screen nonetheless. In particular page breaks are not shown at all, since they are irrelevant to the soft copy.

This is consistent with the philosophy of LaTeX which is to represent the structure of a document rather than the layout. The power of this approach is obvious when you consider that converting LaTeX files to HTML is trivial... try doing that to a random Word for Windows document!

True document preview may be accomplished by running LaTeX over the output of LyX and then using xdvi or a combination dvips and ghostview. This is easier than it sounds, being accomplished by a single pull down menu selection.

In fact LyX is useful even for jaded LaTeX jockeys since the pull down menus cover nearly all of the common LaTeX functionality. This ease of use makes this most powerful of all typesetters accessible to even the casual user. Even complicated structures such as tables become straightforward.

Naturally the dedicated hacker can embed raw TeX and LaTeX directives into the document to produce special effects. In addition there is a mechanism to define one's own style and template files.

So much for the silver lining. The cloud is that installing LyX was a significant undertaking. Firstly you need all the support tools: TeX, LaTeX, dvips, xdvi, ghostscript, ghostview and ispell. Then you need the binary distribution of a user interface library called Xforms; it is free, but there is no source distribution. Xforms requires the Xpm library (why, oh why, isn't it part of the standard X11R6 distribution?). Finally you need the LyX source, itself.

The first time I put LyX together, it just simply crashed upon startup no matter what I did. Several hours of investigation pointed the finger at the XForms package which eventually turned out to have a major incompatibility with Xpm-3.4g (the current version). Arrrrrgggggh. And no chance of fixing it since there is no bloody source. Binary distributions are evil.

Once more around the block with Xpm patched down to 3.4f yielded a working LyX. Well, a very buggy, kinda working if you looked at it sidelong LyX. Now to be fair this is clearly labeled as BETA software so a quick check back to the ftp site gave me version 0.8.2 instead of 0.8.

Nirvana at last. LyX is not yet perfect, but just wait until it gets out of beta test. In the meantime it is quite reliable and effective in producing great looking documents, while still being the best thing since sliced LaTeX. Better still, it satisfied Linda which means I get to play more Doom.

To balance the karmic total, Lyx definitely has some inherent weaknesses. Its Achilles heel is the satanic binary-distribution-only Xforms package. LyX is also a major project to install, simply because of the sheer number of software packages that make up the working whole.

If you are interested, you can find out a lot more on the LyX homepage (http://www-pu.informatik.uni-tuebingen.de/users/ettrich). I'd be mighty interested to hear about anyone else's experiences with this tool.❖

Miscellanea

as spotted in some cheesy newsgroup by pmc, who claims no responsibility for its parentage!

I shot a query into the net.
I haven't got an answer yet,
But seven people gave me hell
And said I ought to learn to spell;

A posted message called me rotten
For ignoring mail I'd never gotten;
An angry message asked me, Please
Don't send such drivel overseas;

A lawyer sent me private mail
And swore he'd slap my ass in jail—
I'd mentioned Un*x in my gem
And failed to add the T and M;

One netter thought it was a hoax:
"Hereafter, post to net dot jokes!";
Another called my grammar vile
And criticized my writing style.

Each day I scan each Subject line
In hopes the topic will be mine;
I shot a query into the net.
I haven't got an answer yet...❖

Background:

Information Security System responsibilities, structure and development

Markku J. Saarelainen

Is your vital business information safe or are you just assuming that this information is safe? Have you established an adequate Information Security System (ISS) to protect your key information against unwanted external or internal visits and use? The changes in the usage and utilization of the information technology have created new requirements for both the information management and its security.

However, still too often businesses and companies do not take the information protection seriously enough to establish proactive information security systems and other controls. If some controls have been established, these controls often focus primarily on the physical security instead of the company-wide information security.

The business information such as business plans, market strategies, trade secrets and others is a very valuable organizational asset, and it would be foolish not to initiate adequate security controls to protect this key asset within the whole organization including physical facilities, employees, external contractors, computers systems, contract negotiation processes and any other business process.

Who is responsible for the information security? Everybody. However, the extent of this responsibility varies from one function to another or from one person to another. Fundamentally, the top management including the organization's CEO is responsible for establishing the information security system.

The top management is responsible for defining, documenting and comunicating the company-wide information security policy to all levels of the organization. In addition, the executive management may establish either specific or general information security objectives to transform the organization from one situation to a more protected situation. The executive management is also responsible for appointing the Information Security Officer (ISO), who performs and acts as the Management Representative and has the authority and responsibility to establish, implement and maintain the information security system.

All other members of the organization are responsible for implementing the information security policy in their daily activities. Some individuals may have additional responsiblities such as ISS auditing and monitoring in accordance with the documented and planned information protection arrangements. The top management is responsible for reviewing the performance and suitability of the system periodically to ensure its suitability and any need for revising the policy, objectives or the system itself.

The structure of ISS is unique to each organization. The responsibilities and authorities are different in all systems, because organizations are unique. However, there are some general requirements that can be used to design and develop the unique ISS for any organization, but still meet basic and fundamental information protection requirements.

These requirements can include all or some of the following main categories:

- Management Responsibility
- Client / Customer Contract Security
- Information Systems Design and Development
- Document and Data Control– and Configuration Management
- Purchasing Information Security
- Facility Management and Physical Security
- Information Systems Management
- Information Security System Audit
- Personnel and Employee Security
- Legal Information Security Matters
- Counter Information Security System Activities
- Information Security Insurance Administration

Each of these general categories have more detail and specific requirements including both documentation, activity recording and data control requirements. Using these requirements and any guidelines, the business can establish its unique information security system that protects the integrity of the information effectively and accurately.

The information security system has to be designed and then developed to eliminate any potential security risks. This requires planning and proactive thinking. The development can start from the Information Security Policy and Objectives that is developed by the

executive management. After this the completed system manual can be developed by the Information Security Officer. This ISS manual should make reference to all applicable additional procedures and instructions that are used within the system.

Typically, these procedures (such as Information Security Disaster Plan and Procedures) described WHOs, WHATs, WHENs, WHEREs and in some cases also HOWs such as back-up instructions and methods. If it is necessary, additional security plans can be developed for any specific project or process. These plans should be consistent with an overall ISS. The masterlists or other equivalent methods should be developed and maintained to control all ISS plans and documentation.

The planning of the information security system provides an excellent opportunity for the management to evaluate and analyze all information risks and design practical and useful approaches to eliminate these risks.

Nobody should underestimate the need for the ISS, but this need should not be artificially created either. The information security system as any system has to be practical and really bring tangible benefits. This is one reason why the information security assessment should be performed prior to the development project. This assessment can identify both weaknesses and strengths in the information security.

Careful evaluations can help the business to focus on real issues, and not to develop the system that meets some imaginary requirements, but fails to address those key areas and functions of the organization, where additional controls would really be needed.

The information security system should be developed for the management, but the ISS users do also include all employees within the organization - and as in many other organizational development, the complete implementation of the ISS shall be dependent on the employee security and their awareness.

Background:

OOPS and GUI: Object Oriented Programming Systems and Graphical User Interfaces

Lawson Hanson

The recent shift in thinking away from the well worn functional description paradigm, and towards that of object oriented trains of thought, represents an attempt to focus on the common complexities that engulf our software projects these days. Systems such as Air Traffic Control, Stock Market Futures Investment Management, National Freight and International Cargo Scheduling, Manufacturer Resources and Production Planning, and hoards of other similarly complex projects have been able to benefit from this so called: 'object oriented paradigm shift'.

In the past we used to specify a project in terms of functional points, that is, each separate function that was to be performed by the software was described in minute detail so that a technical expert could come along and write a technical specification to describe how the function was going to be implemented in some particular computer programming language system. The main trouble with this approach is that it tends to defer any thought about the actual implementation until very late in the design cycle, and we find that ad-hoc data structures seem to grow like topsy to cope with the communication of notional objects between the project's modules and functions.

Event Driven Software

The advent of the X Window System brought with it the requirement to think about computer programs in terms of the range of events that are possible in a windowed environment with a mouse driven pointer. In addition to dealing with input from devices such as the keyboard, and from computer files, the user is able to move the mouse pointer into and out of windows (rectangular regions bounded by frames which may contain control buttons), and may pop-up or iconify windows, select text widgets (directing keyboard input to a specific region of the screen), activate pushbutton widgets, select items from menus, set the

value of scale widgets, select items in a list widget, and adjust the position of a horizontal or vertical slider bar to control the visible portion of a list. These, and many other events can be detected and acted upon in an X Window System program; hence the method of specifying and documenting such programs must be able to cope with each of the possible actions that the user may decide to take, as well as being able to describe the function that is to be performed when the user takes the action. Essentially the structure of such describing documents needs to deal with items on a screen by screen basis, and on a widget by widget basis within each screen.

Object Classification and Functional Ownership

The main idea of the object oriented approach to program design is to start at the other end of things, and to think about the various 'objects' with which the program will deal, and to organise, and group together the objects into classes of similar items. In this way, the inherent complexities of a project can be dealt with first, instead of leaving that to the technical programmer, or implementor, the idea is that we sit down first to identify what it is that we are intending to do with the project; that is what is it that we want to achieve, and what are the visible objects upon which we wish to operate. Once we have identified these, it is easier to begin to see the relationships between these entities, and any commonality can be dealt with to allow the complex items to be decomposed in terms of the less complex.

Instead of the objects being implemented by ad-hoc data structures, or files, or arrays, or lists, etc., at the time of implementation, the object oriented approach allows us to identify the objects first, to classify the objects to find commonality, and then to describe the functions that 'belong' to these objects and their super-sets and sub-classes.

The human brain is usually incapable of dealing with any more than a few items of complexity at any one moment, and so this approach allows us to cope with items that are enormously complex in the fine detail, by considering them to be composed of a collection of simpler objects with which we can deal.

In a sense, the principle of the decomposition of the complex into the simple has been applied in functional terms for decades, but the object oriented approach allows us to shift the ownership of the complexity; that is, in the past, a complex project was considered to be composed of a set of interacting functions which just incidentally manipulated a series of objects about which we initially thought very little; where as from the object oriented approach, a complex project is considered to be composed of a set of interacting objects, each of which we have thought about very carefully, and have designed and assigned a set of specified, object oriented functions --- a set of functions that is oriented to the specific requirements of the object in the project. Hence, the functionality is owned by the object, and the object (complete with its specified functionality) forms a part of the overall project.❖

Background:

One System Administrator's story

Glenn Huxtable

A recent legal case in the U.S. should set off warning bells for all professional systems administrators.

In November 1993 Randal Schwartz was the target of an FBI raid on his home. Within two hours he was dismissed from his employment at Intel, and was later convicted on three computer crime charges.

Randal Schwartz is not a cracker. He is an educator, the author of two respected computing texts, and had been a consulting systems administrator to Intel for over five years. He had simply been doing his job - or so he thought.

Randal started at Intel in 1988. in what was to become their Supercomputer Systems Division (SSD), where he recommended that they maintain basic computer security by following some standard procedures, including using strong passwords. He started checking the strength of passwords in mid-1991 by running a common security tool called Crack.

Crack is a program familiar to most systems administrators. It checks a set of encrypted passwords for common words found in dictionaries, and combinations such as dictionary words with a letter replaced by a number. It is a tool that can alert the systems administrator to poorly chosen passwords that are easy ways in for a cracker.

In late 1993, while working in another division at Intel, Randal wondered whether the systems administrators back in his old division were still scanning the password files.

Using his account on one of the SSD's computers, he ran Crack and found one password straight out of the

dictionary. Fearing the problem was more widespread, he followed the insecure account to another computer within the Intel network, and confirmed that it used the same insecure password. He then collected 600 more passwords from other computers on Intel's network and found that 48 were insecure.

What Randal had discovered was that someone breaking into one of Intel's computers could follow these passwords from computer to computer across their network, picking up new password files on each computer and potentially breaking more passwords that might lead to still more computers.

At the same time, a fellow systems administrator noticed the long running Crack programs and reported them to his manager, who rather than approaching Randal, reported it up the chain of command.

Three days later, before Randal could collate his findings and report his recommendations, the FBI called on his home and confiscated all his personal computers. He was dismissed from Intel within two hours of the raid. Intel apparently thought they had found a corporate spy.

Randal had been given accounts on Intel computers for the purpose of administering Intel systems. He had overstepped his responsibilities by running crack on another division's computers, even though he believed he was doing so the best interests of his employer.

One would expect him to have been called into the managers office and asked what he was doing.

Instead he was charged under a vague Oregon State law against 'altering' and 'transporting' computerised information. One charge of 'altering' referred to creating a file that forwarded his electronic mail to another computer outside Intel when he was travelling on other business.

Two charges of 'transporting' referred to copying password files from one of Intel's computers to another. This was considered stealing the information, even though it never left Intel.

He was convicted on all three counts.

Randal made mistakes. He admits as much and has publicly apologised to Intel on many occasions. He should not have presumed to run Crack on another division's computers, for which he was no longer responsible. He should have reported his activities at the outset, instead of waiting until he had compelling evidence of the security problems. He should have told his fellow systems administrators his suspicions and enlisted their help on the machines for which they were responsible.

Systems administrators jobs are almost always poorly defined, with little demarcation of the scope of their activities. Consultants often work under even vaguer job descriptions. Systems administrators often have to feel out their own limits. Intel isn't unusual in having no clear guidelines about using their systems.

This case sends a clear message to all of us. We need to be very careful about how we go about our work, especially in areas where corporate policy is not clearly defined.❖

Background:
Microsoft Network

Phil McCrea

Earlier in the year it seemed that a right royal battle was looming between the Internet and users of the imminent Microsoft Network (MSN). Hardly an issue of the popular IT press went by without some article or other on how the Internet would be swamped by the Microsoft Network.

But it's all gone quiet recently. What has happened? With the wisdom of hindsight, let's take a brief look at the forces that have been at play over the past year or so in the on-line services industry.

It's important to realise that the on-line service industry has been much larger in the US than it is in this country. Americans have been regular users of on-line services for quite a few years, and providers such as Compuserve, America on Line and Prodigy had been vying for market dominance. Compuserve is the only such service available in Australia, and it's available through Fujitsu Australia. IDC reports that around 25% of US home computer have modems, whereas the figure here is around 6%. I've seen other figures that differ somewhat, but they all point to the much greater use of on-line services by Americans.

It was the potentially lucrative on-line service industry that lured Microsoft in this direction: after all, Microsoft all but owned the desktop (Netscape wasn't even a glint in Jim Clark's eye at that stage...). When the Microsoft Network was hatched, Microsoft had Compuserve in its sights. Furthermore, the on-line service industry is transaction based, and this would provide Microsoft with an alternative business model: the notion of being able to gather a small percentage of each network transaction was (and still is) a compelling one.

I'm fairly sure Microsoft did not really regard the Internet as competition at the beginning. It was perceived as being used mainly by academics who aren't big spenders, and who were not, in general, users of Microsoft software (at the time anyway) - they were UNIX users! In any case the Internet had a lousy user interface which required some technical understanding of UNIX to be able to use it properly.

What Bill Gates did not foresee was the creation of the World Wide Web. In fact, none of us did! But I'm fairly sure that if Gates had been aware of the effect the Web would have, his strategy for the Microsoft Network would have been quite different.

I still recall the first time I heard about the Microsoft Network. It was about two years ago at a lunch when a well known Microsoft Australia employee stated that we needn't concern ourselves with the Internet in the future - Microsoft was bringing out its own network which would supersede it...

And so the Microsoft Network was born. In Australia a joint venture with Telstra was established, a unique situation in the world for the Microsoft Network. Earlier this year, On Australia were touting MSN as being a competitor to the Internet, with the line: "Why would you trust your corporate information to the Internet when you can use the security of MSN". I actually heard these words used! At that stage there was a plan for integration with the Internet, but it would happen over a two year timeframe.

For quite a while corporate Australia was swept along with the MSN euphoria, but the bubble was burst by two different events. Firstly, George Colony from Forrester Research made a whirlwind visit to Australia midyear, and was well reported in the press for his views on how proprietary networks would soon give way to Internet based services. Right on cue, Compuserve, Prodigy and America On Line announced that they would separate service from delivery mechanism, and that they would be using the Internet for delivery.

The second event was of Microsoft's own doing; they almost bought Intuit, the creator of the world's most popular home banking software. In other words, Microsoft could well have become a funds transfer agent, and as a result could have been competition to the financial institutions they were targeting as customers for the MSN. More than one bank has confessed to me their concerns in this area. As a matter of interest, Intel did something similar several years ago when it came out with its own PCs, upsetting the PC suppliers it was selling processors to.

Over the past few months, Microsoft has changed its view of the Internet. So much so, in fact, that when it was launched in the US recently, MSN was provided with Internet access. This is not the case in Australia yet, as it is still being delivered by Austpac. Current projections are that it will be Internet based here early next year.

Microsoft's original strategy was to build up content that was to have been unique to MSN - again emulating the Compuserve model. But this has been less than successful, and very few organisations have been willing to place their content exclusively with a single service provider, particularly when that service provider still has a very fledgling network. Furthermore, the Microsoft network relies on users having access to Windows 95, which will still take some time to penetrate the corporate market. Another concern for some Australian organisations is that the information on MSN resides at Microsoft's corporate headquarters in the US.

Again, several financial organisations have told me that they are quite prepared to be part of the Microsoft Network, but not exclusively: i.e they want to be accessed by their customers who are already users of the Internet.

So how do we view MSN now? Essentially it has become another Internet Service Provider, competing with dozens of others operating in Australia at present. Personally I welcome this: it is a victory for Open Systems - and that's what AUUG is all about.❖

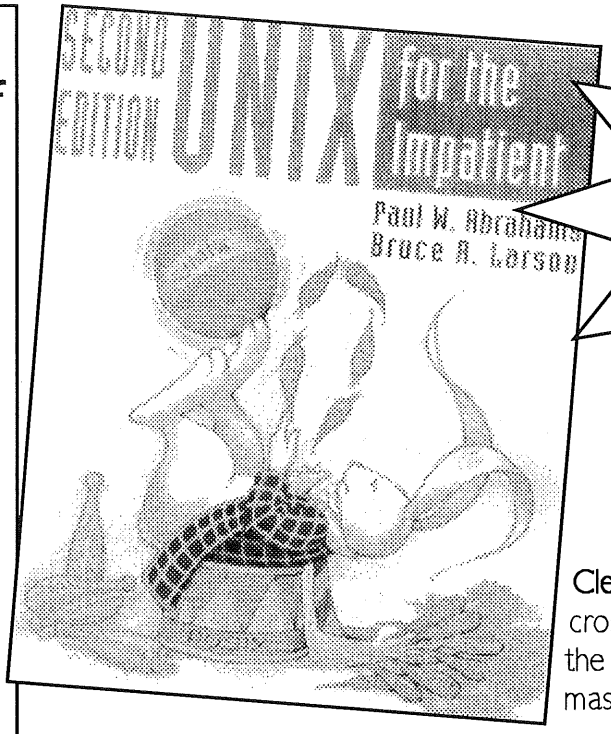
New edition of one of our bestselling UNIX books now completely revised & updated.

UNIX FOR THE IMPATIENT

Second Edition

Paul Abrahams and Bruce Larson

0-201-82376-4
650 pages/ Paperback



20% off for AUUG members

Clear, concise, and readable. Thoroughly cross-referenced and indexed, UNIX for the Impatient is the power-user's guide to mastering UNIX quickly.

Organized functionally and useful for any UNIX implementation on any platform, including System V, BSD, LINUX, Solaris, POSIX.2, ULTRIX, UNIXWARE (Novell), and others.

Covers **all essential information** including:

- File operations
- Data manipulation using filters
- The awk programming language
- Shells and shell scripts, now using the KornShell
- Editors — vi, ex, ed, and GNU Emacs
- Mailers, remote addressing, telnet, ftp, uucp, WWW, other Internet issues
- GUIs, including X Windows

I would like to order _____ *copy/copies of* ***UNIX for the Impatient*** *by* Paul Abrahams and Bruce Larson (82376-4) at the special AUUG price of $35.95 (RRP $44.95)

❑ *I enclose a cheque for* $ _____ **OR** ❑ *Please charge my credit card No:* _____

❑ BANKCARD ❑ VISA ❑ MASTERCARD ❑ AMERICAN EXPRESS ID NO: _____

EXPIRY DATE: / / AUUG MEMBERSHIP NO: _____ PHONE NO: _____ (WK HRS)

NAME: _____ SIGNATURE: _____

STREET ADDRESS: _____

_____ STATE: _____ POST CODE: _____

Send your order to ➤

Addison Wesley Longman

Unit A1, 6 Byfield Street,
North Ryde, NSW 2113, Australia.
Telephone (02) 878 5411
Customer Service Fax (02) 888 9404

LONGMAN

ADDISON-WESLEY

UNIX Tricks & Traps

Edited by Janet Jackson<janet@dialix.oz.au>
Phone/Fax (09) 295 4753

I need more submissions for this column. So next time you answer a question for someone, make a note about it and send it in (or just forward me the email). If one person wanted to know, chances are others will too.

Merry Christmas and Happy New Year to all UNIX users, whether novice, intermediate, guru or wizard.❖

Tracing system calls

Gregory Bond <gnb@bby.com.au>
and
Janet Jackson <janet@dialix.oz.au>

`trace/strace/truss` is your friend!

Don't ya hate it when some weirdo piece of software won't run, and all it says is "file not found"—but won't tell you which bloody file?

Run it under `strace` and Lo! the answer!

(Useful for all manner of weird undocumented situations, too.)

(Feel free to embellish this to taste!)

OK, editor's embellishments: on most UNIX systems there is a program called `trace`, `strace` or `truss`, for tracing the system calls that a running process is making. You either run the program under `trace` (or whatever), or do something like `trace -p PID` to trace an existing process. `trace` prints (in overwhelming detail) each system call the process makes.

I've found this most useful for debugging processes that hang. Is it waiting for a child process? waiting for a read? or what? If it is in fact waiting for a system call, not looping in user mode, `trace` will tell you.

To find out more, say `man trace` (or whatever) on your own system.

Unfortunately my BSDI BSD/OS system doesn't have a `trace`. But it comes with almost all the free software I want, from `less` to `LaTe`, pre-installed, so, as Marika used to say, I no complain.

Now here's a story about a time when I found `trace` useful.

Trap: interpreters can't be scripts

Janet Jackson <janet@dialix.oz.au>

About a year ago I discovered something I didn't know about `execve()`, the system call that loads and runs a new program.

The site had a homemade setup for locally installed software, something like the following. Users ran such software out of `/local/bin`; the `/local` hierarchy was NFS-mounted across all hosts. `less`, for example, was found in

```
/local/bin/less
```

but this was a symbolic link to a shell script

```
/local/bin/start.sh
```

This script determined what UNIX flavour the host was running—say HP-U—and ran the appropriate binary from a path something like

```
/local/hpux/bin/less
```

This worked okay until I installed Perl under this scheme. Perl is an interpreter: its purpose is to run scripts. Say I had a Perl script called

`testscript` in the current directory, and this script started with the usual "hashbang":

```
#!/local/bin/perl
```

Most of you probably know that when `execve()` sees the "`#!`" at the start of the file, it knows it has to load the interpreter named after the "`#!`" (`/local/bin/perl`), and give the file to that interpreter as input (instead of just loading and running the file as a normal executable).

I found that

```
perl ./testscript
```

worked fine. However,

```
./testscript
```

which is supposed to make `execve()` behave as just described, didn't. As far as I recall, the system tried to run the script under `/bin/sh`, which is used as the interpreter in situations where the system can't tell which one to use.

After a good deal of work, I figured it out, and sent the following mail to a colleague:

Subject: the hash-bang problem is fundamental

I'm fairly sure I see what's going on now. It is the way the system call `execve()` is implemented. `execve()` does not recurse.

Given

```
execve("./inner.csh", argv, envp)
```

(where `argv[0] == "./inner.csh"`, etc), `execve()` notes that `./inner.csh` is a script, so it should "invoke" (as the manpage fuzzily puts it) the interpreter. However, it does not do another `execve()`

to do this invocation—yo
a `trace`. It simply tries to
so if the interpreter is not
error.

Remember, our interprete
and this was a symbolic li
`/local/bin/start.sh`. T
in a.out (executable) form
`execve()` responded to tl
interpreter by running `/b`

Apart from the actual trap
things to be learnt from th

Firstly, as the manpage is
than code, you can't alwa
had assumed that "invoke
`execve()`". (I guess it's jus
you might end up with ar

Secondly, the `trace` progr
previous article was a use
suspicions.

But most importantly, this
of stripping a problem do
that no extraneous factors
for this is "a controlled exp
there might be something
I had installed. Then I thou
in the code of `start.sh`, b
Eventually I devised a con
execute a simple script un
that took Perl and `start.s`
found that the same probl
seemed a quite different si

AUUG NSW Chapter Summer Conference 1996

"The Commercial Internet Comes Of Age"

Blue Mountains, Feb 28-March 1, 1996

CALL FOR PAPERS

We all now know what a wonderful, if not essential, gestalt entity the Internet is. We read every day about how we must get connected or perish.

But, after the Internet has found a cure for cancer, rediscovered Lasseter's Reef and invented a perpetual motion machine, some serious questions will remain: What, in my business, will I use the Internet for?

Do I really want to be connected, and why? If I regard a World Wide Web page as a good form of advertising, should I be surprised when my employees spend their whole day surfing the Web? If I do want to connect, how should I go about it?

If you can answer any of these questions, or if you have more questions of your own, you are invited to submit a paper for the AUUG NSW Chapter Summer Conference 1996.

We also seek papers on topics related to UNIX and Open Systems.

TUTORIALS

We seek tutorial presentations on topics related to the Internet, UNIX and Open Systems. A day of tutorials will precede the conference, and half day or full day tutorials are invited.

LOCATION

In order to better serve those members of AUUG NSW Chapter who live outside Sydney, we will be holding the Summer 96 conference in the Blue Mountains.

The Conference will consist of 1 day of tutorials and two days of paper presentations.

SUBMISSION GUIDELINES

Those proposing to present papers should submit an extended abstract (1-3 pages) and a brief biography.

Those submitting tutorial proposals should submit a plan for the tutorial and a brief biography.

Please indicate on your proposal whether it is a paper or tutorial, and for tutorials whether they are half day or full day.

PROPOSALS SHOULD BE SENT TO:

AUUG NSW Summer 96 Program Committee
PO Box 6425
North Sydney, NSW, 2060
Phone: David Purdue on (02) 906 3766
Email: auug-nsw-exec@auug.org.au

IMPORTANT DATES

Abstracts/Proposals due:	1 October 1995
Authors notified:	1 November 1995
Final copy due:	1 February 1996
Conference:	28 Feb-1Mar 1996

Paper:

Attention to detail!

by Janet Jackson <janet@dialix.oz.au>.

(First published in the Proceedings of the Sixth Annual Canberra AUUG Conference and Workshops, February 1995.)

Abstract

While experience, intuition, and seat-of-the-pants hacking are essential to the work of UNIX systems administrators and programmers—and they know it—what we also need for professional-quality work is attention to detail.

Whether we are talking about producing bug-free, maintainable code, securing networks, preventing crises, or troubleshooting complicated systems, we need workable documentation, procedures and standards that ensure consistency without stifling innovation.

In this paper I discuss my ideas on how to do this sort of thing right, based on my experience, and give examples from a wide range of real-life situations.

1. Introduction

While experience, intuition, and seat-of-the-pants hacking are essential to technical UNIX work, what we also need for professional-quality work is attention to detail.

Plenty has been said recently about ensuring quality at the organisational level; I will focus on what can be done by the team and the individual, on their own initiative.

This paper makes various suggestions and gives many examples for both software development and systems administration. These are not meant to be comprehensive—if they get you thinking even a little bit, this paper will have achieved what I want.

2. Basic attention to detail for software developers

Software quality assurance is a large field in which I am no expert. It is also a term that strikes fear into the heart of programmers, who, being creative and independent types, don't like being told how to do their jobs.

However, truly professional work takes more than just great code. There are some basic techniques that should be as much a part of programmers' repertoires as the programming languages they use, whether or not they are employing some methodology or organisational standard.

These are mostly things I didn't learn at uni or read in textbooks (although both going to uni and reading textbooks can be helpful). I've picked them up on the job over the years. Having to maintain and install other people's software has taught me that attention to detail is needed from the moment you touch the keyboard. It can't be added at the end.

Having to maintain my own code six months or more after I wrote it has taught me a lot too.

2.1 Basic documentation

Every program should come with some documentation, even if the programmer is its only user.

Firstly, document what the program is for. The name of the program is rarely enough of a clue. Even if you are the only user, when you come back to it in six months time it will probably not be as obvious as it is today. And if other people have to use it... their minds don't work like yours, so your brilliant naming idea probably means nothing to them.

This doesn't always mean you have to write a manpage. For your personal scripts, a description at the top of the file may suffice. For a directory of programs used by a small team, a README file in that directory may be enough. However, under UNIX, programs to be used by general users should *always* have a manpage, even if it is trivial. It is very frustrating to say `man mysterious` and get the response `No manual page for mysterious`.

Secondly, document how to run the program. What arguments does it require? What options does it have? Do you have to be root? and so on. This information should go in the same place as the information about what the program is for. For shell scripts I usually put something like

```
USAGE="Usage: $0 [-v] [filename..]"
```

in the first few lines of the file, where `head(1)` can find them. `USAGE` may also be used in error or help messages.

When documenting options in manpages, make the options easy to find. Write a proper OPTIONS section with the options used as paragraph labels. Like this:

OPTIONS

-b Batch mode. Don't ask for interactive confirmation.

-d Debug mode. Display reams of information.

Please don't do this:

The program asks for interactive confirmation unless you give the -b option. If you give the -d option, it will display reams of information.

This is unpleasant enough for human readers, but some people have programs to automatically find options in manpages. These rely on the labelled OPTIONS format.

Make sure your manpage includes sections on FILES, if any are used, and BUGS (which have, regrettably, been renamed to NOTES or RESTRICTIONS by certain insecure vendors lately). Even if your BUGS section says "no bugs are known at this time", that's useful information.

And lastly, document how to compile the program. For programs that need to be compiled, it seems obvious to me that you should document how this must be done, but I've seen many programs without such documentation. It's as if the programmer thought it would be a good challenge for his or her colleagues to have to work it out.

For a single C file, you can use a comment like

```
/* Use cc -o thing thing.c -lm */
```

near the top of the file. But for anything more complex than that, take the time to write a Makefile [6]; it will save yourself and your successors a lot of trouble.

2.2 Attention to detail when coding

Much has been said about coding style, change control, and testing.

Cannon et al [1] is an excellent paper on C coding style; Straker [10] has written a whole book on it, including a chapter on the psychological reasons why good coding styles work. For discussions of change control, testing, and quality assurance in general, see, for example, Frakes, Fox and Nejmeh [2] or Spuler [9]. Pomeranz [7] gives some guidelines for Perl [11] style.

I am not going to discuss indentation, identifier names, modularisation, portability and so on, except to say that it is important to maintain good coding practices, even when writing little throwaway programs. After a while good practices become habits, so that it is no extra work to apply them when they are really needed.

I am not going to discuss change control and testing either, except to say that from what I have seen, more is often needed! (In my first programming job, my boss forced me to keep a logbook of changes—about the best thing she ever did for me.).

However, I am going to discuss comments.

Even the best programmers seem to have a cavalier attitude to comments. Presumably they understand their code perfectly and therefore assume it is self-documenting. *There is no such thing as self-documenting code*, unless you count trivia like `echo hello world!`.

Now, I don't go in for formal loop invariants and such: as far as I'm concerned they're just more code. However, there's a minimum level of commenting without which understanding is difficult.

1. A comment at the top of each source file, describing the purpose of the file and the objects (functions, types, and so on) therein. I've seen many a source file containing screensful of copyrights, author details, and even change logs, but no indication of what the module does!

2. A comment at the top of each function or subroutine, no matter how trivial, saying what it does, how it should be called, what it returns, and what its side-effects are.

3. A comment beside the declaration of each parameter, constant or variable (I'll make an exception for loop counters, but that's about all) saying what it is for.

4. Beside or above any complicated or obscure construct, an English or pseudocode description of what it is supposed to do. This is especially important where there are boundary conditions.

5. A comment at the closing brace (or whatever) of each subroutine and each block longer than about 10 lines, saying which construct is ending. The same goes for `#ifdefs`.

2, 3 and 4 are particularly important for maintainers because they explain what the code is *supposed* to do. This is not always the same as what it does! And without the comments, the poor maintainer can't tell what the original programmer intended.

Put in your comments as you write the code, not afterwards. If you leave them until the end they'll never get done. And having them available during development and testing makes it easier for you to understand your own code.

The usual objection to commenting as you go is that you then have to change the comments when you

change the code, and this is too much trouble. My answer to this is, basically, shut up and get on with it, because it's worth the trouble. And if you'd used comments from the start your code might work better (because the comments improve your own understanding) and might not be needing so many changes.

So get into the habit! Eventually you will learn to think in both Perl (or whatever) and English at the same time, comments will come as naturally as code, and those who have to read your programs will be eternally grateful.

3. Attention to detail for systems administrators

Much has been written about software style, but very little has been written about systems administration style.

Systems administrators, have you considered your style? What about the style of your team (if you're lucky enough to be in one)? How does it feel to you? How does it look to your clients (users, managers)? Is it professional?

I believe that a good systems administration style is a combination of big-picture planning and attention to detail.

3.1 But where to start?

In systems administration, there are a huge number of areas where attention to detail is needed. So where do you start?

You start with backups. If you have adequate, reliable backups, *anything* can go wrong. But it's not as simple as just running `dump(8)` once a week and stashing the tape in your desk drawer. Plan your backup schedule carefully to suit your site's needs. Make sure the backup and recovery procedures are documented. Verify your backups—don't trust that they worked. (See [13] for an exploration of ways backup programs can fail.) Keep several sets of backups in case your verification wasn't enough. Keep a known good set of backups off-site, with a copy of the recovery procedure, in case of a fire in the machine room or similar.

To further improve your chances of disaster recovery, document the location of your distribution media, and procedures for reinstalling from scratch. (This can be useful even when there has not been a disaster—yet. How many times have I had to frantically hunt for manuals to tell me how to boot single-user, or how to reinstall boot blocks on a root partition, because that machine-specific information wasn't kept near the machine or with the backup tapes?).

Your disaster recovery kit should also include the contact details of critical staff and vendor support people, and colleagues from whom peripherals, cables, media, and so on can be borrowed in emergencies.

3.2 And after that?

Your next area of attack will depend on what is most important to your organisation. The following are a number of areas where I think attention to detail is particularly useful.

Client service. Client service, also known as user support, is an area where a little attention to detail can make a huge difference.

Keep your users informed about what you are doing with the systems. Remember that the systems are there so that they can do their work, not so that you can try out the latest software or techniques. Users may not find it easy to understand what you do, let alone see how your work is helping them—so you have to tell them. Establish a newsletter or an online bulletin, or simply email round an update now and again. (Don't do it too often or people get hassled by it. Once a fortnight is a good place to start.).

If you're doing direct user support (help-desk), try to give a sense of personal service. I have found that setting up an email "hotline" address (called `support` or `help` or the like), and sending individual, personalised replies, helps a lot. Taking the trouble to set up form letters for common queries can save a lot of time and anguish. Remember to leave room for `Hi $NAME` at the beginning and a personal signature at the end so the user knows he or she is dealing with a human.

When you do deal with people face-to-face or over the phone, don't forget to smile. (It makes you feel better too.) Try to take an interest in the person's work, but don't overdo it or you may create unrealistic expectations. And try to remember people's names. Gabor [4] describes a number of tricks for this.

A technique I have found handy is to mentally associate each user with the last thing I helped them with. Then, when I see them again, I remark, "How's the XXX going?" It isn't much trouble, and I have found very few people who are not pleased by this kind of attention.

Wilson [12] has written a good book on providing technical support.

Security. It should be obvious by now that security is important if you are on the Internet or have dial-in modems; but even if you're not, internal security is still vital. There are few organisations where everyone is so trusted, and so competent, that accidental or intentional violations of data confidentiality or integrity never occur.

And attention to detail—vigilance—is what systems and network security are all about. I'm not going to discuss all the issues here; read Russell and Gangemi [8] for a detailed introduction to computer security, and Garfinkel and Spafford [5] for practical information about securing UNIX systems. Meanwhile, start with passwords, because if they are weak, other security measures are a waste of time.

Crisis prevention and record-keeping. You will always get some crises, and how you handle them will often determine whether they recur.

I have not yet mentioned that it is a good idea to **keep logbooks**. You might have one logbook recording everything you do; one per computer; one for hardware faults and one for software changes—whatever works for your site, but *keep records* of some kind. Why do I think this is such a good idea?

- If a problem recurs, you can look back and see how you fixed it. It's helpful if the logbook has some kind of indexing, even if it's only Post-It tape flags.
- It helps you remember where you're up to, and helps stop you forgetting small details. (*Did* we install that patch?).
- It covers you against future allegations.

If you change a system file, record what you have done, and save the original with a suitable extension (`/etc/fstab.old` for example) in case you make matters worse. For some files, such as `sendmail.cf`, it's a good idea to always keep the vendor original (in `sendmail.cf.dist`, for example), as it may come in handy when you're installing patches.

One more thing about recording what you have done: do not assume that you will be the only person who ever touches the system. You might not get run over by a bus—but you might get a new team member, or you might move to greener pastures and be replaced. And it's no good for you, the new person or the organisation if they're constantly having to ask questions like, "How come `/bin/login` has such a recent modification date? Have you patched it? Have you replaced it? Where's the old version?".

Keep a central directory to record patches. For miscellaneous changes, either keep a list in a well-known location, or create files such as `/bin/login.README` recording what happened. At the very least, this makes sure *you* don't forget why you did things.

Creating, automating and documenting procedures. Systems administration involves a lot of routine tasks that, once recognised as such, can be documented, and often automated. Some obvious candidates are adding new users, installing new machines, running regular backups, security checks and other systems housekeeping, and restoring files for users.

Such routine tasks often involve many small details that must not be forgotten. These are often site-specific, and so are not mentioned in the vendor's documentation. For example, when a new user is added he or she may have to be added to a `staff` email alias. When a new machine is installed, there are usually many site-specific post-installation steps that should be done before the users are let loose on the box. If such details are written down, or better still, coded, there's a much better chance that they will get done reliably, even in the face of staff turnover.

In some organisations, documented or automated tasks can be delegated to operators, help desk people or clerical staff, freeing the systems administrator to concentrate on the more specialised tasks, such as further improving the systems.

If you can't delegate tasks to people, then delegate them to the computer. In my experience, you should try to make the time to automate anything that can be clearly defined—and for tasks that must be done regularly, have `cron(8)` run them silently and only alert you when there are problems. Sometimes this seems a lot of trouble initially, but it is usually made worthwhile by the time you gain and the tedium you lose.

3.3 The importance of planning

From time to time you will have to make a major system change, like installing a new `sendmail(8)`, upgrading an operating system or repartitioning a disk.

I have seen many people who merely assign a night (such things usually have to be done at night), tell the users (some don't even bother with that!), then turn up and start working. Only to find that they don't have enough scratch tapes. Or that step X needed to be done *before* step Y, but it's too late now. Or that it's going to take 24 hours, and the users are going to be upset, but we're beyond the point of no return. Or that the whole thing just isn't working, but there's no easy way to

back out, so we have to try to sort out the mess *now*, under pressure.

There is a way to avoid all these situations. It's called planning. Before you assign the night or tell the users, figure out exactly what you have to do (write down all the steps, and review them to make sure nothing is missing), estimate how long it will take, determine what materials you will need (how many tapes, for example) and get hold of them, and write a backout plan (this may be as simple as restoring the whole system from tape, or it may involve undoing changes to various files).

In your planned steps and your time estimate, always include a backup—and back up even those parts of the system you don't expect to alter, just in case. Before you make any changes, verify that your backups are restorable.

A written, detailed plan will take time to prepare, but it will be time well spent. You (and your boss!) can be reasonably sure that, barring unforeseen bugs, the change will go OK and be done in the available time window; and if it doesn't, you will be able to calmly bring the system back to its original state. Either way, you will come out looking very professional.

4. But doesn't disorganisation promote creativity?

Is too much attention to detail, too much organisation, a bad thing? Aren't disorganised people more creative?

Disorganised people may *look* more creative, but I don't have any reason to believe they have more or better ideas than organised people. I don't know of any statistical studies in this area.

In my own experience, too much organisation, where it becomes an end in itself, can stifle or distract the mind; but the right amount of organisation frees the mind. Documentation and automation provide a kind of extension to your memory, into which you can offload the detail, leaving your "central processor" free to think about larger matters. Big picture thinking is important too.

For example, once you have written your Makefile, you no longer have to remember how to compile your program, and are free to concentrate on its design.

Or in systems administration, once you have got your user-adding procedure sorted out, you will have gained some time in which to create user training resources, or whatever else is important.

5. Conclusion

If UNIX programmers and systems administrators are to be professionals, doing work of consistently high quality, they must combine big-picture thinking with attention to detail.

Software developers must get into the habit of taking care with their documentation, change control, testing, coding style and commenting, even in small projects.

Systems administrators should consider their style, too. Attention to detail in backups and disaster recovery is imperative. Other areas, such as client service, security, crisis prevention, and routine tasks, can benefit from a detailed style. When making major system changes, a step-by-step plan is a professional's secret weapon.

I hope that the suggestions and examples I have given will make you think about how you can make your own work more professional.

References

[1] Cannon, L W, et al. Recommended C Style and Coding Standards. `//ftp.cs.washington.edu.au:/pub/cstyle.tar.Z` (1990).

[2] Frakes, W B, Fox, C J and Nejmeh, B A. *Software Engineering in the Unix/C Environment*. Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1991.

[3] Free Software Foundation. GNU project software. Available from, eg, `ftp://archie.au/gnu` or `ftp://prep.ai.mit.edu/pub/gnu`.

[4] Gabor, D. *How to Start a Conversation and Make Friends*. Sheldon Press, London, 1985, pp 83-92.

[5] Garfinkel, S and Spafford, G. *Practical Unix Security*. O'Reilly & Associates, Sebastopol, California, USA, 1991.

[6] Oram, A and Talbott, S. *Managing projects with make*. O'Reilly & Associates, Sebastopol, California, USA, 1991.

[7] Pomeranz, H. "Perl Practicum: A Plea for Clarity" *;login:*, 19(6), December 1994, pp20-21.

[8] Russell, D and Gangemi, G T, Sr. *Computer Security Basics*. O'Reilly & Associates, Sebastopol, California, USA, 1991.

[9] Spuler, D A. *C++ and C Debugging, Testing and Reliability*. Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1994.

[10] Straker, D. *C Style: Standards and Guidelines*. Prentice-Hall, Hemel Hempstead, UK, 1992.

[11] Wall, L and Schwartz, R L. *Programming perl*. O'Reilly & Associates, Sebastopol, California, USA, 1991.

[12] Wilson, R. *Help! The Art of Computer Technical Support*. Peachpit Press, Berkeley, California, USA, 1991.

[13] Zwicky, E. "Torture-testing backup and archive programs." *Proceedings, Usenix Large Installation Systems Administration Conference (LISA V)*, San Diego, California, USA, 1991, pp181-189.

Book Reviews

Frank Crawford <frank@ansto.gov.au>

Well, another book review section has rolled around very quickly, in fact almost too quickly, and it's the final one for the year. This lot are exclusively O'Reilly and Associate books, some of which have been around for some time, other of which have recently been revised.

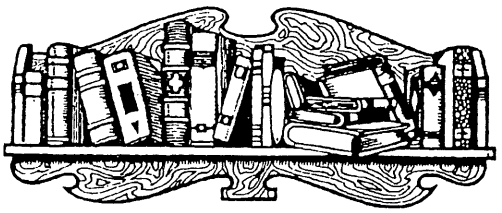
Out of interest, I've now sent out my 100th book for review, and I've generated some interesting statistics, one of which is a break down by publisher. The totals by publisher were:

- O'Reilly and Associates: 39
- Prentice Hall: 29
- Addison Wesley: 20
- others: 12

As many of you have noticed, we currently have lots of books coming for review. The current practice is to post a note to the mailing list **auug-books@ansto.gov.au** and the newsgroup **aus.org.auug** when we have new books available. Unfortunately, this disadvantages members without network connections, or on the end of a low speed link. For people in such a position, either mail, via the AUUG PO Box, or fax me on (02) 717 9273, with your contact details and preferences.

As the mailing list seems to be proving popular, I am planning to expand it next year, so keep an eye out for announcements, and if you aren't already subscribed to it, do so.

Finally, have a good Christmas, and watch out for more books next year.❖



Programming with GNU

by Mike Loukides
O'Reilly & Associates, Inc. 1995
ISBN 1-56592-112-7
Reviewed by Alex Kowalenko
Alex Kowalenko Associates
<alex@mpx.com.au>

What would UNIX be without the Free Software Foundation? The Free Software Foundation, and the GNU tools have become synonymous with UNIX. Many system administrators and developers use the GNU tools on a day-to-day basis. They provide an alternative to the commonly used tools and utilities supplied with modern UNIX installations, and in some cases they are shipped standard with the UNIX installation. They also provide an alternative when some UNIX tools are now being "unbundled" from the operating system.

But what is the Free Software Foundation? What is GNU? Quoting from the bulletin of the foundation:

> *"The Free Software Foundation is dedicated to eliminating restrictions on people's right to use, copy, modify and redistribute computer programs. We do this by promoting the development and use of free software. Specifically, we are putting together a complete, integrated software system named GNU that will be upwardly compatible with UNIX."*

The FSF are moving towards this goal, with most of the common UNIX utilities already available. Their UNIX-like kernel - HURD, will soon be available to people who what to alpha test the system. As a side product, most of the tools developed for GNU have already been available for other UNIX installations.

Many a time, developers face the problem of obtaining tools with which to develop. The budgets of software managers tend to be very small for tools, or there is a lack of understanding of why it's needed. "We bought you a X thousand dollar C++ compiler last month, and now you want a debugger? You should be grateful for a compiler! ..." The GNU tools have often used to plug the hole, so and ease a developers burden, at no cost to the project managers budget.

I was interested to see what tools would be covered in the book. Since the FSF aims to provide a free alternative to all the common UNIX utilities, there could be quite a wide range of packages to discuss. Also some of the tools tend to have minimal documentation provided, usually as reference manuals, so learning how to use the packages effectively is sometimes an effort.

There are a wide variety of tools from the FSF, but 'Programming with GNU' covers tools related to UNIX software development. The packages covered include: emacs, gcc, gdb, make, RCS, libc & libg++,

various binary utilities and GNATS (bug tracking system). The book also includes an guide to installing the FSF software, and an introduction to UNIX.

'Programming with GNU' provides a good introduction to programming in the UNIX environment. Most of the tools are copies of commonly used tools UNIX development tools. Topics covered include compiling a C program, using make to manage compilation of some C files, using gdb to debug a program. Some one with little knowledge of UNIX development would find the book a good start for programming.

For experienced developer, the book tends to fall short. I found the chapter on Emacs interesting and learnt some new things from it (I think you can never ever stop learning a few new keystrokes in emacs), but I found some of the other chapters a bit thin. The book is clearly aimed at beginners, the chapters on libraries (libc and libg++) and on GNATS would cover as an introduction. More information would need to be provided to make good use of the tools.

O'Reilly already publish books on some of the packages including Emacs, Make, SCCS & RCS, and programming UNIX with the C language. There were some packages I would like to have been seen included like DejaGnu (automated testing), and CVS (similar to RCS but can work on whole directories of files at a time), but that probably would been outside the scope of Mike Loukides's intentions.

The book comes with a CDROM, with the source code to all the GNU tools discussed in the book. Also included are binaries for the following platforms : SUN Sun-OS 4.1.3 & Solaris 2.4; HP 9000/700 HPUX; IBM RS/6000 AIX 3; SGI Iris/Indigo Irix 5; and Intel x86 Linux, plus some additions from Cygnus support, who provide commercial support for these tools.

O'Reilly & Associates have a strong reputation for producing quality guides to various software tools. I think that "Programming with GNU" matches to that standard. I would recommend the book to anyone starting to program in the UNIX environment.❖

Essential System Administration 2nd Edition

by AEleen Frisch
O'Reilly and Associates, Inc. 1995
ISBN 1-56592-127-5
Reviewed by David Baldwin
ANU
<David.Baldwin@anu.edu.au>

The book gives a good overview of UNIX system administration. It appears to be targetted towards smaller "traditional" installations of a handful of UNIX machines, concentrating mainly at the level of a single machine.

With the exception of network security, subject areas relating to large installations, complex services and desktop integration are largely ignored. While this is largely beyond the scope of a reasonable sized book the emphasis on older technologies such as directly connected serial terminals and modems while ignoring networked devices such as terminal servers, network printers and X-terminals (eg use of BOOTP and TFTP for booting) is a deficiency since such devices are now a common part of many UNIX installations.

The book is a good reference source with well worked examples and diagrams. There is good coverage of the philosophy as well as practice of system administration, and philosophical digressions on approachs and alternatives to practice and application are scattered throughout the text.

There is good coverage of various contemporary dialects of UNIX including SunOS, Solaris, AIX, IRIX, OSF/1, HP-UX, SCO and Linux. Many examples using proprietary tools and utilities are included. The book assumes a fairly high level of UNIX sophistication - eg a working knowledge of utilities such as 'grep', 'awk', etc. It adds good discussion of applications of these tools and utilities such as 'find' and 'xargs' to administrative tasks, and issues fundamental to transition from expert user to sysadmin - eg file system organisation, system startup and shutdown.

The user account administration section includes extensive discussion of password security, user environment and proprietary account management tools. The security section has a good discussion on vendor supplied and freely available utilities, including 'tripwire', 'COPS', etc. There is no discussion of problems with reuseable passwords beyond chosing "hard to crack" passwords. In particular there is no mention of freely available one time password schemes such as S/Key which are rapidly gaining acceptance. Proprietary one time schemes are mentioned in passing. There is good detail on vulnerabilities and things to check for traces of break-ins or security problems.

The system resources section gives a good overview of system resources and how they interact and how to gain some understanding of system and process resource usage. There is good discussion of various quotas and their effects. The automation chapter covers use of freely available tools such as 'perl' and 'expect' (but not 'tcl'). There is gruesome detail on connecting serial terminals and modems. The printing section covers queues and remote printing between

UNIX systems, but assumes printers are always physically connected. Many sites use printers directly connected to their network and not directly to a print server. The networking section is not comprehensive enough to cover the full complexities of TCP/IP network admin, however common daemons and services such as NFS are covered reasonably well, although network booting of devices is not covered at all, although there are examples of disabling 'tftpd' as being a security hole! NIS is glossed over and NIS+ is mentioned but in passing. Security tools such as `tcp_wrappers` are covered and firewalls are briefly mentioned but only with a high level functional description. Much of this area is covered in greater detail in other publications.

The book is well layed out with chapter headings giving overview of material covered. The copy reviewed was an early draft and there were many omissions of diagrams, typographical errors, etc which are probably corrected in the final version. There was no index in the draft copy.

Overall a good reference source covering many details of the management of a UNIX system including good coverage of vendor-specific idiosyncracies, but with some deficiencies in areas of integration into a complex network environment.❖

Running Linux

by Matt Welsh and Lar Kaufman
O'Reilly & Associates 1995
575 pp, $49.95
ISBN 1-56592-100-3
Reviewed by Ian Crakanthorp
ANSTO
<ian@ansto.gov.au>

Over the last few years the growth of Linux, a free implementation of UNIX, has been quite phenomenal. Used in business, education, and the home, it provides a complete system with a large number of utilities and applications that can run under it.

Being the resident Linux-head at work, I first ran Linux on a 286 PC when it was version 0.11, basically all you got was a dollar prompt, so this book called Running Linux was of interest.

Running Linux gives a full introduction and general overall guide to Linux. It is aimed to provide enough information for both UNIX novices and more experienced users and I think it achieves this. It would be a very handy reference for anyone using Linux or thinking of using Linux for the first time. As such it is not a book that needs to be read cover to cover.

The contents of each chapter of the book are as follows:

Chapter 1 provides a history and introduction to Linux. A comparison of Linux to other PC operating systems is given. Hardware requirements, supported hardware is discussed, plus a whole section on where to obtain even more information about Linux.

Chapter 2 is a well written guide on how to obtain and install Linux. Including how to boot, create partitions, swap space, file-systems etc. And a whole section on problem solving if difficulties are encountered.

Chapter 3 presents a basic tutorial on UNIX.

Chapter 4 is a basic guide to systems administration on a Linux host. Including modifying system configuration files, compiling the kernel, adding users and the other many things system administrators do.

Chapter 5 discusses a range of UNIX utilities, what they are, how to use them, how to install and configure them if necessary, and some brief tutorials. The utilities include vi, emacs, Tex/Latex, X-Windows, email programs, DOS tools, and multi-media tools.

Chapter 6 discusses programming with Linux. Using gcc, make, perl, shell scripts, Tcl and Tk. Each with a short guide and tutorial. A short description of other languages and utilities is given. Then a whole section on gdb, followed by a range of other programming tools including gprof, strace, RCS, patch and others.

Chapter 7 is a guide to networking and communications with Linux. TCP/IP PPP, SLIP, the World Wide Web, email, Kermit, UUCP are all covered.

Appendix A contains a list of Online documentation for Linux.

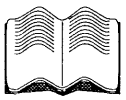
Appendix B contains a vendor list, no Australian ones though.

Appendixes C and **D** contain information about FTP sites and Bulletin Board sites respectively.

Appendix E contains the GNU General Public license that Linux falls under, for those interested.

The book contains a lot of useful information about 'running Linux'. Some sections about the applications are not Linux specific and could easily be used as a guide for other non- Linux UNIX systems. The writing style is clear and easy to understand. Anyone thinking

of installing Linux or just wanting to know about Linux will find this a very handy reference.❖



Making TeX Work

by Norman Walsh
O'Reilly & Associates, Inc. 1994
483pp.
ISBN 1-56592-051-1
Reviewed by Jagoda Crawford
Australian Nuclear Science and Technology Organisation
<jc@ansto.gov.au>

TeX is a typesetting system, as opposed to the widely used WYSIWYG word processors, such as Word Perfect or Microsoft Word. TeX has the same general paradigm as troff (the author refers to troff as being the "other" text formatting system), i.e. they process a plain text file with control sequences and produce a typeset document.

Donald Knuth designed TeX in the late 1970s and it is increasing in popularity, particularly in typesetting mathematical equations, scientific text and multiple languages. Further, many technical journals accept articles submitted in TeX.

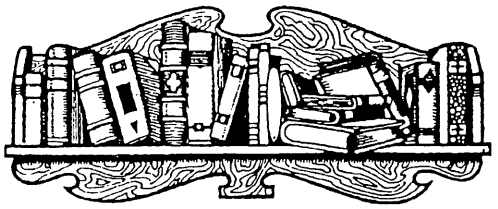
The TeX system is not a single program, but a set of programs, tools, fonts and a number of related files. Two programs form the core of the TeX typesetting system. One of them is TeX itself, the program that reads your input files and transforms them into typeset form. The other major program is METAFONT, a tool for creating fonts. Producing TeX documents involves a series of steps, including editing the document, running TeX itself, and processing TeX's output in various ways.

"Making TeX Work" surveys the world of TeX software and describes how the various pieces are related. As such, it is of benefit to system administrators interested in learning enough about these programs to install and test them for their users, and for users to see the "big picture" and the variety of utilities that are available. Although, people intending to use TeX for their document typesetting will need at least one other reference describing document preparation.

The book contains the following:

- the various tools available for use with TeX, which are summarised in an appendix for quick reference,
- how to get TeX and its associated tools from public domain and commercial sources,
- how to assemble the software you need to build and install TeX on common platforms- such as UNIX, OS/2, MS-DOS, Windows and Macintosh,
- how to install common macro packages. There are a large number of them and choosing the ones appropriate for local use is very important,
- description of the importance of fonts and how to install and use fonts to best advantage,
- how to select and use the tools that let you incorporate graphics into your documents. A survey of a number of packages for manipulating images is given,
- how to create bibliographies using various bibliography database tools and how to incorporate citations and references in your document,
- how to create indexes,
- how to run TeX and find errors using the log files, how to setup your editor to run TeX and spell checking tools, and
- a chapter on International Consideration is also given.

I found the book very useful as it outlines the whole structure of TeX and its related programs. I recommend it as a reference book for a site using TeX and particularly when first starting to use TeX. Reading this book should result in the right choices being made during the first installation. On the other hand, it will not be the only reference as it doesn't cover (or even try to cover) the details of text formatting.❖



WAAUG news: From the Western Front

Edited by Tom Hallam
<thallam@geol.uwa.edu.au>

I'd like to start this column by thanking Janet Jackson for the excellent job she's done of editing the Western Australian section of AUUGN over the last few years. Janet has been editing the WAUGs news letter since October 1991 and has now decided to hand it over to someone else. As Janet said when she asked for volunteers "Other things are starting to get in the way of it, and anyway it's about time someone else had a go." Not that we've heard the last of Janet, she's still editing "UNIX Tricks & Traps" and I'm sure (I hope) that she'll be an active supplier of articles.

This month we have reports from our "Meeting Reporters". As always I'm on the lookout for articles from anyone who's willing to write them.❖

From the Chair

Adrian Booth <booth_adrian@tandem.com>

Having just moved into an apartment in a "security complex" has had me thinking about the difficulties we all face in determining the right balance of security versus convenience. Take the rear door to the carpark - you need to pull the handle and turn the latch at the same time to open it. Of course they have been situated just far enough apart that you need two hands to do it, which is wonderful when you are carrying a briefcase or some rubbish to put in the bin (try moving house \fI\into\fi\ a security complex sometime!).

I sort of wonder what the architect had in mind when it was considered essential to have to unlock a door to get \fI\out\fi\ of the building! It is hard to see how is could be classed as a security "feature"! Its interesting to consider myself a "user" of the building and experience what it is like to have to deal with a security implementation over which I have no control and no understanding. As far as I can see all it does is get in the way of what I want to do!

I guess there are several lessons for people involved with securing anything. First and foremost, determine what you are protecting, and from what threats. Second, determine how much it is worth protecting in terms of time, money, effort, and inconvenience (Fort Knox is very inconvenient to get gold out of but I guess we can all understand why). And thirdly—much of the hassle we have with "secure" systems today is because we are taking inherently unsecure systems and trying to build a secure system from them... a bit like trying to turn a PC running DOS into a functional network workstation! It can be done - but I would rather have an as-designed system sitting on my desk than what amounts to a kludge.

So what else has been happening in W.A.? Briefly...

Planning for the summer conference is well underway. Janet Jackson, the program chair, has received many abstracts from interstate and even overseas, but only a handful locally. This year's conference should be a very cosmopolitan event!

WAUG members who attended our last meeting at the Blue Dive will recall that we ended up going across the tracks to another venue and ordering in a large pile of pizzas. This went down so well that the committee is considering having a big pizza night for the December meeting! Whether we can swing that at the Freeway Hotel is open to debate but I for one am looking forward to it! :-)

We are starting to see some fresh faces at our meetings again, which is great. Early next year I hope WAUG will be in a much better position to actively promote its existence to the W.A. computing community, and so hopefully we will see that trend continue!

Note sure exactly when people will be reading this, but compliments of the season to everyone, and in Janet Jackson's words, "...may your kernels never panic in the New Year"!❖

Meeting Reports

September

UNIX in the Real World

Presented by: Luigi Cantoni
STM, <lui@stm.com.au>

Meeting Reporter: Tom Hallam,
UWA, <thallam@geol.uwa.edu.au>

At our September meeting Luigi Cantoni gave a light hearted but informative talk on "UNIX in the Real World" based upon his experience with supplying computer systems to the fast food industry. UNIX is often though of as being a complicated operating system which requires a high level of computer competence to manage. In the "real world" these

systems need to be used by people with the minimum of computer skills.

His talk started with a brief description of computer jargon from the users point of view. For example, the box in the corner is the "floppy drive" and the thing which displays letters is the "computer": we'd normally call them computer and screen! (It's all a lot clearer when you are waving the said item around the room!) As you can imagine this level of ambiguity can cause some communication problems especially when you are trying to fix a problem over the phone line. Turning the screen of when you've told them to turn the computer off just does not work!

Luigi then went on to describe some of the problems he's had to deal with and how they were fixed. I related well to the tale of the lightning strike which took out all the screens and serial ports, I'd seen it before. And the two headed modem cable which was plugged into parallel port as well as the serial port reminded me of a printer I'd seen plugged into a SCSI port. Yes the modem is plugged in and turned on. Then why isn't it working?

There was a brief discussion off some of the design issues for building reliable and easy to use systems. This covered such things as the layout of the keyboard, the choice of serial boards and modems. The use of modems for providing support, upgrades and data transfer.

This was a light hearted and enjoyable meeting attended by the few people who had not gone east for AUUG'95. It demonstrated that UNIX can be used to provide a reliable, easy to use solution to real world problems.❖

October

WWW Security Pitfalls

Presented by: Don Griffith,
Curtin University, <griffith@cs.curtin.edu.au>

Meeting reporter: Frederick W. Myers,
Diamond State Data Services,
<fredwm@perth.dialix.oz.au>

While most of us have (and had) heard of the World Wide Web, Don's talk covered a number of features which should be on the minds of every security or systems administrator (or UNIX system owner).

Most browsers cannot [yet ;-)] handle the various multimedia formats which make the WWW so enjoyable to use. The ability of browsers (and the CGI Common Gateway Interface) to easily add new interpretors may make for easy use but even Ghostscript can have a program (read web page) fed to it which can access data from the receiving machines hard drive. Trojan or embedded programs in otherwise innocent data are a favorite method for hacker entry. The bottom line is that you should know where all your browsers have had their source. Microsoft Word has its macro language - an implemation of BASIC. So be aware of the capabilities of your innocent programs.

Another source of problems is "leakage" of data from your system - that is data other than what you want sent off-site. Many browsers have the standard capability of having the user "fill in forms". Many servers also query the browsers for general capabilities. While it serves a purpose in defining what can be handled by the user, the data, in the wrong hands, could be misused. Two examples; Joe uses Browser X which has a known hole, therefore the server machine knows exactly what steps to bypass security. Secondly if you use Browser Y perhaps you might receive unsolicited mail for "add-on" products, while you wonder how they knew about your use of the base product.

But what can the user and administrators do? Prevent (or restrict) the loading of helper programs onto the browser. Use a proxy server to isolate networks from the main Internet traffic. Educate users about the traps inherent in Net communications (One simple way is to bring them to AUUG chapter meetings). Learn more about what other people have encountered. Monitor the Internet newsgroup comp.security.announce.

If your want to run a server, run it as a screened subnet, better yet isolated as a single machine. Run the server as a non-privileged user. Perhaps a non-UNIX-based machine would limit the capabilities of those who might actually gain access. Finally don't run anonymous FTP and WWW server on the same machine.

For those wanting more information the following books and web-pages were recommended:

Chapman & Zwicky: *Building Internet Firewalls* O'Reilly 1995

Lui et al.: *Managing Internet Information Services* O'Reilly 1994

The World Wide Web Security FAQ **http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html**

Safe CGI Programming **http://www.primus.com/staff/paulp/cgi-security/safe-cgi.txt**

Other WWW site references to security **http://www.cs.purdue.edu/homes/spaf/hotlists/csec.html**

Many thanks to Don for an excellent talk and to all those who came to listen. With a few more perhaps we will need to look for a larger room.❖

Meeting information

WAUG meets at the Freeway Hotel, 55 Mill Point Road, South Perth. We meet at 6:15pm on the third Wednesday of each month.

Our meetings are advertised in the Diary column of the Computers section of Tuesday's *West Australian*.

If you need further information about the next meeting, please contact Mark or one of the committee.

SPEAKERS ARE NEEDED

especially ones who can actually commit to giving a talk on a certain date! So if you can give a talk, or know someone who can, please let us know. Mark (our meeting organiser) cannot produce them out of thin air.

WAUG Email Aliases, Newsgroups and Web Page

WAUG has the following mail aliases on **uniwa.uwa.edu.au**:

waug-membership - for membership enquiries
waug-chair - our Chairperson
waug-meetings - our meeting organiser
waug-secretary - our Secretary
waug-newsletter - for newsletter contributions or enquiries
waug - for general correspondence (will be read by the Secretary, as a paper letter would be).

So, for example, you may send general correspondence to **waug@uniwa.uwa.edu.au**.

Check us out on the World Wide Web at: **http://www.auug.org.au/auug/waug/waug.html** (thanks Canberra AUUG).

Also see the newsgroups wa.waug and aus.org.auug for announcements and discussion.

Committee Contact Details

Office-bearers:

- Chair : Adrian Booth 321 9111
  booth_adrian@tandem.com
- Treasurer: Patrick Ko 483 8111
  pko@DIALix.oz.au
- Secretary: Major 357 5076
  major@yarrow.wt.com.au

Ordinary committee members:

- Mark Baker (Meeting Organiser) 491 6081
  baker@telecomwa.oz.au
- David Buck
  dbuck@ncc.telecomwa.oz.au
- Luigi Cantoni 474 3700
  lui@DIALix.oz.au
- Don Griffiths 351 7691
  griffith@cs.curtin.edu.au
- Tom Hallam 380 2665 (AUUGN Sub-editor)
  thallam@geol.uwa.edu.au
- Glenn Huxtable 328 8288
  glenn@fs.com.au
- Janet Jackson 272 5061
  janet@DIALix.oz.au
- James Patton (Meeting Reporter)
  jrp@mrwa.wa.gov.au

A publicity officer has not yet been appointed. Please volunteer. (Adrian is acting in the interim, but don't you think he has enough to do already?)

For Systems Administrators: Local SAGE-AU Meetings

The WA Regional Group of the Systems Administrators Guild of Australia (SAGE-AU) meets on the *first Tuesday of each month at 6pm, in room G3 at the Alexander Library* (Note the change).

There will be no meeting in November.

The next meeting will be on Tuesday 5 December 1995. It will consist of a talk by Glen Huxtable on the recent Randal Schwartz legal case and the AGM.

If you manage computer systems for a living, we'd like to have you along. SAGE-AU is **NOT** another UNIX group. All systems and network administrators are welcome. We would particularly like to see more PC network administrators attending, so if you know any, send them along. I'd like to see lots of Novell, NT, OS2 and MAC people attending.

For more information, please contact any of the following:

- regional group chair,
  Janet Jackson <janet@DIALix.oz.au>,
  (09) 272 5061
- meeting organiser
  Mike Horton <mikandfi@DIALix.oz.au>,
  (09) 479 8424
- Tom Hallam <thallam@geol.uwa.edu.au>,
  (09) 380 2665

For information about SAGE-AU in general, you may also look at **ftp://ftp.sage-au.org.au/pub/SAGE-AU** and **http://www.sage-au.org.au:8080/**.❖

AUUG Canberra news:

Canberra chapter

John Barlow, 019 935477,
<cauug.secretary@auug.org.au>
Secretary, Canberra Chapter of AUUG Inc.

AUUG Inc Summer Conference, February 1996

We are organising the Summer conference (February next year) and need volunteers to help with organisation, PLUS we want people to offer presentations to be given for the conference PLUS we want people to offer workshops for the workshop section.

If you have any comments, suggestions or requests - please come forward and make them now, so we have a chance of organising what you want.

The INTERNET Project

If you are an AUUG member in the Canberra Chapter you can get email and news access to the Internet, and there is no cost (so long as you remain a financial member). This is provided by a dialup service which has been recently updated and has a direct IP connection to the Internet. Full IP access to the Internet can be obtained at extra cost ($120 for 400 hours, approximately). If you are interested in this service, please contact John Barlow (mobile: 019 935477) to have a chat about it.

We have a Linux box and a FreeBSD box available via the same dialup service, so if you want to examine these great (free, full source) UNIX implementations, here is a prime opportunity ! Existing users who wish to use these boxes need to email linux@auug.org.au or freebsd@auug.org.au to have an account setup.

Meetings organiser wanted:

If you want a specific topic discussed at a future meeting, or want a specific UNIX presentation made, please contact John Barlow (contact details at the end of this message).

Coming Events:

December 12

Christmas drinks at the "cellar bar", university house, ANU

January

NO MEETING

February

Summer conference and tutorials

AUUG Institutional Members

as at 22/01/96

AAII
ACAY Network Computing Pty.Ltd.
Actrol Parts
Adept Software
Alcatel Australia
Amalgamated Television Services
Amdahl Australia
Andersen Consulting
ANI Manufacturing Group
Ansett Australia
ANSTO
Anti-Cancer Council of Victoria
ANZ McCaughan
AT & T GIS
Attorney-General's Department
Ausnet Services Pty. Ltd.
AUSOM Inc.
AUSTA Electric Qld Minerals & Energy Centre
Australian Archives
Australian Bureau of Statistics
Australian Centre for Remote Sensing (ACRES)
Australian Customs Service
Australian Defence Industries Ltd.
Australian Electoral Commission
Australian Film Television and Radio School
Australian Information Processing Centre Pty. Ltd.
Australian Medical Enterprise
Australian Museum
Australian National Audit Office
Australian National University
Australian Submarine Corporation
Australian Taxation Office
Australian Technology Resources (ACT) Pty. Ltd.
Australian Technology Resources Pty. Ltd.
AWA Defence Industries
B & D Australia
Barwon Water
Bay Technologies Pty Ltd
BHP Information Technology
BHP Information Technology
BHP Minerals Exploration
BHP Research - Melbourne Laboratories
BHP Research - Newcastle Laboratories
Burdett Buckeridge & Young Ltd.
Bureau of Meteorology
Bytecraft Pty. Ltd.
Cape Grim B.A.P.S
Capricorn Coal Management Pty. Ltd.
CelsiusTech Australia
Central Queensland University
Central Sydney Area Health Service
Centre for Open Systems Pty. Ltd.
CITEC
Clegg Driscoll Consultants Pty. Ltd.
Coal & Allied Operations
Cognos Pty. Ltd.
Com Net Solutions
Com Tech Communications
Comcare Australia
Commercial Dynamics
Commercial Industrial Computer Services Pty. Ltd.
Communica Software Consultants
Composite Buyers Ltd.
Computechnics Pty. Ltd.
Computer Associates
Compuware Asia-Pacific
Continuum Australia
Copper Refineries Pty. Ltd.
Corinthian Engineering Pty. Ltd.
CSC Australia Pty. Ltd.
CSIRO Division of Information Technology
CSIRO Division of Manufacturing Technology
Curtin University of Technology
Cyberdyne Systems Corporation Pty. Ltd.
Cyberscience Corporation Pty. Ltd.
Cybersource Pty. Ltd.
Daedalus Integration Pty. Ltd.
Data General Australia Pty. Ltd.
Datacraft Technologies
Dawn Technologies
DB Bain Group Services Pty. Ltd.
Deakin University
Defence Housing Authority
Defence Service Homes
Department of Communications and the Arts
Department of Conservation & Natural Resources
Department of Defence
Department of Defence (TC Section)
Department of Education QLD
Department of Family Services & Aboriginal & Islander Affairs
Department of Gaming & Racing
Department of Lands Housing & Local Government
Department of the Treasury
Department of Urban Services
Dept. of Industrial Relations Employment Training & Further Education
DEVETIR
Dialix Internet Services27
Digital Equipment Corp. (Australia) Pty. Ltd.
Dominion Systems Pty. Ltd.
DSTO Lab 73
EASAMS (Australia) Limited
Edith Cowan University
Electricity Trust of South Australia
Electro Optics Pty. Ltd.
Engineering Computer Services Pty. Ltd.
Environmental Resources Information Network (ERIN)
Deparment of Environment Sport and Territories
Equity Systems Pty. Limited
Ericsson Australia
ESRI Australia Pty. Ltd.
Execom Consulting
Executive Computing Group
FFE / James Hardie Bldg. Serv.
FGH Decision Support Systems Pty. Ltd.
Financial Network Services
First State Computing
Flinders University
Fremantle Port Authority
G.James Australia Pty. Ltd.
GEC Alsthom Information Technology
Genasys II Pty. Ltd.
Great Barrier Reef Marine Park Authority
Haltek Pty. Ltd.
Hamersley Iron Pty. Ltd.
Hannan Group Computer Services
Heath Insurance
Hermes Precisa Australia Pty. Ltd.
Hitachi Data Systems
Honeywell Australia Ltd.
Honeywell Ltd.
Hong Kong Jockey Club Systems (Australia) Pty. Ltd.
I.P.S Radio & Space Services
IBM Australia Ltd.
Ideas International Pty. Ltd.
Independent Systems Integrators
Informatel Online
Information Technology Consultants
Insurance & Superannuation Commission
Integration Design Pty. Ltd.
Intelligent Network Development
James Cook University
Joint House Department
JTEC Pty. Ltd.
Keays Software
Knowledge Engineering Pty. Ltd.
Laboratory Systems Pty. Ltd.
Labtam Australia Pty. Ltd.
Land Information Centre
Land Titles Office
Leeds & Northrup Australia Pty. Limited
Logica Pty. Ltd.
Lotus Development
Lyons Computer Pty. Ltd.
Macquarie University
Main Roads Western Australia
Mayne Nickless Courier Systems
Mayne Nickless Information Tech. Services
Medical Benefits Funds of Australia Ltd.
Memtec Limited
Mentor Technologies Pty. Ltd.
Mercedes-Benz (Australia) Pty. Ltd.
Message Handling Systems
Metal Trades Industry Association
Mincom Pty. Ltd.
Minenco Pty. Ltd.
Mitsubishi Motors Australia Ltd.
Mitsui Computer Limited
Moldflow Pty. Ltd.
Motorola Communications Australia
Motorola Computer Systems

Multibase Pty. Ltd.
Multiline BBS
National Library of Australia
National Resource Information Centre
NCOM Services
NEC Australia Pty. Ltd.
Northern Territory Library Service
Novell Pty. Ltd.
NSW Agriculture
NSW Teachers Federation Health Society
Object Design Pty. Ltd.
Object Technology International Pty. Ltd.
Office of the Director of Public Prosecutions
Open Software Associates Ltd.
OPSM
OSIX Pty. Ltd.
Pacific Star Communications
Peter Harding & Associates Pty. Ltd.
Petrosys Pty. Ltd.
Philips PTS
Port of Melbourne Authority
Powerhouse Museum
Primary Industries & Energy
Process Software Solutions Pty. Ltd.
Prospect Electricity
Pyramid Data Centre Systems
Qantek
QLD Department of Transport
Quality By Design Pty. Ltd.
Redland Shire Council
Renison Golfields Consolidated Ltd.
Rinbina Pty. Ltd.
Royal Melbourne Institute of Technology
SCEGGS Redlands Ltd
Sculptor 4GL+SQL
Security Mailing Services
SEQEB Business Systems
Siemens Nixdorf Information Systems Pty. Ltd.
Smallworld Systems (Aust.) Pty. Ltd.
Snowy Mountains Authority
Software Plus (Australia) Pty. Ltd.
South Australian Co-operative Bulk Handling
Specialix Pty. Ltd.
St. Gregory's Armenian School
St. John of God Hospital
St. Vincent's Private Hospital
Stallion Technologies Pty. Ltd.
Standards Australia
Stanilite
State Library of Victoria
State Revenue Office
Steelmark Eagle & Globe
Sterling Software
Storage Technology of Australia
Sydney Electricity
Sydney Ports Corporation
Systek Corporation Pty. Ltd.
Systems Development Telecom Australia
TAB Queensland
TAFE NSW Information Systems Division
Tandem Computers
Tattersall Sweep Consultation
Technical Software Services
TechNIX Consulting Group International
Telecom Australia
Telecom Payphone Services
Telstra Applied Technologies
Telstra Research Laboratories
The Far North QLD Electricity Board
The Fulcrum Consulting Group
The Roads & Traffic Authority
The Southport School
The University of Western Australia
Thiess Contractors Pty. Ltd.
Thomas Cook Ltd.
TNT Australia Information Technology
Toshiba International Corporation Pty. Ltd.
Tower Technology Pty. Ltd.
Tradelink Plumbing Supplies Centres
Transport Accident Commission
Triad Software Pty. Ltd.
Unidata Australia
University of Adelaide
University of New South Wales
University of Queensland
University of South Australia
University of Sydney
University of Tasmania
University of Technology Sydney
Vanguard Computer Services Pty. Ltd.
Victoria University of Technology
VME Systems Pty. Ltd.
Walter & Eliza Hall Institute
Water Board
WCS Australia Pty. Ltd.
Wesfarmers Limited
Western Mining Corporation
Westrail
Woodside Offshore Petroleum
Workers' Compensation Board of QLD
Workstations Plus
XEDOC Software Development Pty. Ltd.
Zircon Systems Pty. Ltd.

From ;login:

STANDARDS

Reprinted with permission

An Update on Standards Relevant to USENIX Members

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

The following reports are published in this column:

- POSIX.n
- POSIX.1a: System API

Our Standards Report Editor, **Nick Stoughton**, welcomes dialogue between this column and you, the readers. Please send any comments you might have to *<nick@usenix.org>*.

Report on POSIX.n: Working On The Trailing Edge

Last summer, at the Boston USENIX, the talk was all about Encryption. The air was hot with discussion about Phil Zimmerman and PGP, the latest news about RSA, Public Key Encryption, and the Clipper chip. This year, at the September LISA 95, the talk was all about electronic commerce. The standards industry has yet to come to grips with either of these issues . . . within POSIX there was an encryption study group, but let's face it, POSIX.1a is still not a standard, and the biggest thing that now addresses is Symbolic Links, which I first remember playing with in 1981. The POSIX encryption study group has now been dissolved, 9 months after its formation – presumably because the area was not ready for standardization. Why are standards so far behind?

Some people complain that it is the process that is too slow, devaluing the result of the standard even before it is finished. Chuck Severance, whose article on the US Technical Advisory Group (a misnomer if ever there was one!) argues this precise point.

However, there are others who have looked at this, and arrived at the other end of the spectrum; the standards process *should* be slow if it is to have value. Standards based on leading edge technology must, at best, be flexible enough, and insufficiently set in concrete, to allow the technology itself to mature. Competition between differing implementations allows the users to choose the best solution. Sometimes what users choose may be the best marketed, and not the best technically, but that's life.

People who want standards must learn to be patient. The process *is* slow, but it's meant to be! The process forces a balance of users, vendors and academics to agree that what is in the standard is right for all of them. It is an open process, and driven by volunteer individuals. In most cases, some company is picking up the tab, but it is the individuals who actually do the work, not the companies.

Working on the trailing edge may not be glamorous, but it is the right place for standards.

Report on POSIX.1a: System API

Shravan Pargal reports on the July 10-15, 1995 meeting in Nashua, NH

The first meeting of the POSIX.1a working group under the new organization commenced with a plenary meeting of the entire System Services Working Group to elect a working group chair. Both Jay Meyer, chair of the former POSIX.1 working group, and Joe Gwinn, chair of the former Realtime working group, were nominated. After some fine campaigning by both candidates, Jay Meyer was elected chair after a short ballot procedure.

The new system services group now consists of:

- Core interfaces (POSIX.1a),
- SRASS (POSIX.1h),

- Realtime (POSIX.1b),
- Removable Media (POSIX.1k)
- Transparent File Access (POSIX.1f)
- Resource Limits (initially a part of POSIX.1) and
- Checkpoint/Restart (also initially a part of POSIX.1)

Joe Gwinn, as runner-up in the election for Chair, was appointed without contest as vice chair of the new working group.

The POSIX.1a document went through some major reorganization during this meeting. Having experienced significant difficulty in getting the checkpoint/restart and resource limits sections through the ballot process, it was decided to split this work off into two separate, new, projects. There is work to do on these areas, and the ballot group had alerted us to the fact that they weren't yet ready for publication. New Project Authorization Requests (PARs) were submitted to the PAR Management Subcommittee (PMC) for these projects. The PMC agreed to recommend approval of the two new PARs. It was decided that separate ballot groups, separate timelines and separate resources were appropriate for each of the two PARs (and corresponding working groups). The new names for the working groups are checkpoint/restart (P1003.1m) and resource limits (P1003.1p).

The resource-limits and checkpoint/restart sections of the POSIX.1a document will be removed from there and will become the starting point for these new projects.

Other hot topics in POSIX.1a continue to cause long discussions, both in and out of the meeting room. How to deal with trailing slashes seems to be such a regular discussion item that the vice chair has even suggested we add it as a fixed agenda item for every meeting!

The current standpoint on this, reflected in the last draft (draft 12), is to insist that a filename can only have one or more trailing slashes if it is a directory. The words currently talk about "as if a trailing /." were appended. However, this still needs some work, but it is unlikely that the next draft will be any more tolerant of trailing slashes on non-directories than draft 12.

Another hot topic is the handling of group-ids when files are created. System V and BSD systems have totally different models here, and the original POSIX.1-1990 tolerated both. Attempts to settle on the BSD behavior have met with enough resistance that it now seems likely that both will continue to be tolerated.

Almost all this discussion is driven from the ballot process for POSIX.1a, currently trying to resolve issues on draft 12 and get a draft 13 out to ballot around the end of the year. It is a long, slow job, as anyone who has been through a ballot must realize, and most of us are new to it in POSIX.1a!

Sometimes, changes are so deep into history that we cannot fathom the reason for them, and have to start all over again. For example, some requirements on `fflush` were changed between the 1988 and 1990, but no rationale was supplied at the time. What should be the effect of `fflush` on a stream opened for reading? Now we have to write rationale for this action that occurred five years ago!

Open Systems, POSIX, and NT

by Stephe Walli
<stephe@srw.com>

"Feds declare NT 'open system'; UNIX takes a hit"

ComputerWorld news headline, July 31, 1995.

That headline in *ComputerWorld* appeared, along with similar ones in other trade publications, after a US Government bid-protest judgement was handed down in June, 1995. These news articles all claimed, more or less, that the Federal Government had somehow declared Microsoft Windows NT to be an "open system;" that this action was somehow "not right" or "unfair;" that customers would not be able to tell sheep from wolves anymore; and that UNIX vendors had better watch out.

In September, UniForum entered the fray with a *UniNews* editorial (Volume IX, Number 15) that was also forwarded to *ComputerWorld* to "correct" their original article. Unfortunately, this editorial was equally full of speculation and religion, and it got a few facts wrong.

My biggest concern is that a public relying only on trade publications with attention-grabbing headlines might not understand the real issues and might blame or lose faith in well-developed, implemented portability standards such as the ISO C-standard, POSIX.1, and the specifications based upon them such as the X/Open Single UNIX Specification.

I was directly involved in the bid protest trial as an expert witness on POSIX related issues for the defense. (Yes, I worked for *that* side), I have been a long-time member of the IEEE POSIX community and care a great deal about its implementation and use. My involvement in that effort began as an application developer in the commercial world, caring about software portability and platform independence.

In this article I will present the facts about the bid and its protests, describe the judgement based on those facts, and finish with a few opinions and observations on "open systems," POSIX, and NT. The information about the RFP, the proposals, the protests, and the judgement is taken from the decision document. While you may not agree with my opinions expressed at the conclusion, hopefully the facts will clear up some of the confusion brought about by the sound and fury that passed for news regarding this case.

The Facts

The US Coast Guard wanted to procure up to 25,000 new workstations (CGSW III) to take them through the next five to ten years, as they replaced an installed base of CTOS machines (CGSW II). These new machines were to be a combination of desktop workstations, departmental servers, and laptops. A suite of office automation applications (including an electronic mail facility) and an RDBMS were also to be provided.

The body of existing Coast Guard CTOS applications was going to be replaced over time through the use of this RDBMS technology, as these applications are written in a very proprietary manner. (Historically, no use of the CTOS POSIX.1 capabilities appears to have been made.) There was no intention of directly porting the existing application source code. The RDBMS technology was part of the procurement, and there were requirements that it adhere to NIST FIPS 127-1 for SQL. This would allow the re-developed applications (providing they were correctly developed) to be more easily ported to other RDBMSs adhering to the SQL FIPS when necessary in the future.

The RFP described the Coast Guard's desire to move in a standards-based "open systems" direction for future work, taking the definition of open systems from the NIST Application Portability Profile (NIST Special Publication 500-187). To this end, the Coast Guard RFP required that proposed systems adhere to a number of standards for specific components (such as the SQL-based component mentioned above). They specified that each machine proposed provide a FIPS 151-2 certificate (POSIX.1) and that the bid winner commit to expanding their POSIX support by providing POSIX.2 (shell and utilities) and POSIX.4 (realtime interfaces) implementations within 12 months of the effective date of a US Government FIPS in these areas.

The RFP required "GOSIP" networking support, and this had ramifications for the email component of the procurement. I will not comment further on GOSIP, as my area of experience lies with POSIX standards and portability, not GOSIP.

Value-for-Money considerations largely governed the open competition and over-all solution. As with many bids, certain demonstrations were required before the bid was awarded, which in turn required the winner to pass further demonstrations. None of these demonstrations had anything to do with software source-code portability or interoperability.

The Unisys Government Systems Group bid NT desktops and servers. There were two other bidders who made the short list. One bid "UNIX" [1] desktops and servers. One bid "UNIX" servers and different workstations that provided a POSIX.1 implementation.

Unisys was the apparent winner and went on to pass the additional demonstrations.

The companies that did not win protested the award to Unisys. Protestor #1 complained that NT is not a "POSIX" compliant operating system. Protestor #2 complained that NT did not fulfill the "GOSIP" requirements. The bid protest process then ground into action with lawyers and technical experts entering the fray.

The GSBCA (General Services Administration Board of Contract Appeals) merged the two protest actions into a single case as the two concerned the same procurement and involved similar allegations. The protests were amended to include specific reasons why the Unisys proposal did not fulfill RFP POSIX and GOSIP requirements (i.e., why NT wasn't "POSIX" and "GOSIP"), whether the government customer knew what it was doing, whether or not the Unisys proposal failed to satisfy the customer's "open systems" objectives, and certain concerns expressed about the conditions under which the platform demonstrations were executed. Recognize that this is what the protests claimed; the protestors then had to prove they were correct.

Bid protests must be completed in 45 working days. During this intense period of time, about thirty lawyers, almost a dozen technical experts, and numerous factual witnesses from the Coast Guard, NIST, and the various vendors involved, testified to the facts, offered opinions and debated the meanings of terms and terminology in the RFP, the NIST procedures, and various other relevant (and irrelevant) documents, building their respective cases for presentation to a panel of three judges. All the material was subjected to intense scrutiny by many people from different perspec-

1 UNIX is a trademark licensed exclusively through X/Open Company Ltd., and used in conjunction with an X/Open branding program. When I use "UNIX" (in quotes) I am speaking of any one of the traditional operating systems of which there are many proprietary versions and some number of public versions available. Any other use of the "U" word will be appropriately placed in the context of the X/Open specification or brand program. The trade rag articles still don't appear to appreciate the difference between "UNIX" and XPG4 UNIX.

tives. This decision was *not* the idle opinion of a few people arguing "open systems" religion.

The protests were denied on all counts. Here is how this decision was reached, based on the "findings of fact" and discussion from the judges' redacted decision. (GSBCA 13201-P, 13211-P, C3, Inc., and TISOFT, Inc., v. General Services Administration, and UNISYS Corporation, June 9, 1995 – released to the public in redacted form on June 30, 1995.)

NT is a FIPS 151-2 certified system, and as such is a "POSIX-compliant" operating system [particular findings of fact are referenced in braces for those who care]:

- FIPS 151-2 is a document that incorporates the POSIX.1 standard (IEEE Std 1003.1-1990 == ISO/IEC 9945-1:1990) by reference, setting certain limits, and requiring certain optional functionality to be supported. {16}
- The NIST certification process ensures that the results of a run of the authorized FIPS 151-2 test suite are valid. This test suite is not a guarantee of conformance, but contains test cases that cover the functionality as specified by NIST in FIPS 151-2. {35}
- A certificate of validation is awarded if everything is correct. {36}
- There is a component of the NT operating system that provides the POSIX.1 specified functionality as defined by the NIST FIPS 151-2 document. {43}
- The NIST POSIX.1 Conformance Test Suite (PCTS) has been run by an accredited testing lab on NT, and the test suite results were submitted to NIST and validated. The NT operating system received its certificate to this effect. {55, 57}
- NIST does not certify "operating systems" as "POSIX" conforming, neither do they test operating systems. They ensure that systems claiming to provide the functionality described in FIPS 151-2 (and the POSIX.1 standard by reference) do so, regardless of how that functionality is implemented. {41}
- The Coast Guard did not have any reason to constrain bidders (reducing competition) by choosing a particular style of implementation of the POSIX.1 functionality. {20} The POSIX.1 standard itself describes why application writers need not care about how a system provides the functionality. (IEEE Std 1003.1-1990, Section B.2.2.2, descriptions of "hosted implementation" and "native implementation.")
- POSIX.1 does not specify an operating system. It is a specification of an interface to an operating system. {13}
- POSIX.1 was written in a minimal fashion, and it was expected that other "POSIX" standards would specify further interfaces that an operating system might provide. {12, 17} The US Coast Guard has certainly required further support.
- Operating systems that implement the POSIX.1 interfaces are often referred to as "POSIX-compliant" operating systems. {23, 24, 25} (I personally hate this lack of accuracy, but it appears to have become commonly abused terminology in our industry.)

Part of the protest revolved around terminology in the RFP that the critical email and RDBMS applications "run under the POSIX operating system." The simple answer to this in the judges' decision is that these two applications run on NT, the NT platforms proposed had their FIPS 151-2 certificates, and therefore the applications run under the POSIX operating system. This simple logic does not begin to describe the painful debates that went on during the protest litigation. Essentially,

- The language "run under" was used by the Coast Guard to prevent bidders from proposing solutions of these applications that were run using emulation. {31}
- Every one agreed that there were other interpretations of the words "run under" and that a poor choice of words had been made, but there was no reason to doubt the testimony of the Coast Guard representative. {32, 33, 34}
- The words did not mean that these two applications' source code needed to be coded against POSIX, or ISO C or any other specification, as that is a source-code implementation issue for the application implementors, based on their own *portability* needs and choices, and makes no difference whatsoever to the end-user of the application, in this case the Coast Guard.

The issue that raises the most heat in this entire protest is the issue of whether NT fulfills the "open systems" direction of the Coast Guard. Unfortunately, this has been reported as "NT declared an open system." No such declaration was ever made by anyone during this bid protest.

Most of the debate focussed on how the POSIX.1 functionality is accomplished on NT. The protests argued that the POSIX.1 functionality is not integrated in a coherent fashion with the rest of NT, and that it therefore fails to meet the open systems' requirements outlined in the Coast Guard's OSE policy and planning objectives. This protest count was denied in the decision, as follows:

- The protest count attempts to argue the acceptability of the Unisys proposal based on the Coast Guard's OSE policy and planning objectives, not the RFP requirements.

- The Coast Guard is clearly committed to moving in a standards-based open systems direction based on the minimum mandatory standards requirements that were imposed in the RFP.

- The Coast Guard did not expect to accomplish its entire OSE objective at one fell swoop in this single workstation bid.

- There were obvious trade-offs made for competition and cost savings.

- There were no requirements in the RFP that proposed the portability and interoperability be integrated in any particular way or to any specific degree. There was no extra credit given during the evaluation period for the degree of portability or interoperability provided.

- The Unisys proposal did provide the Coast Guard a greater degree of portability and interoperability than they have in their current workstation environment.

In the words of the judges' discussion:

> "Protestors are apparently convinced that the Unisys solution does not promote the Coast Guard's open systems environment objective as effectively as their own proposals would. Consequently, they speculate that the acceptance of the Unisys proposal will have serious long-range economic implications so far as future software conversion costs are concerned. This may or may not be true."

In my opinion, based on the stated problems to be addressed by this procurement, and the chosen application conversion methodology being the RDBMS using SQL, the future application conversion costs will probably be related to how proficient the application development team is with SQL and the RDBMS, rather than anything to do with POSIX.1 and ANSI C. The RDBMS happens to be the NT version of a traditional "UNIX" database, so this should further aid the applications' portability to other platforms supporting this particular RDBMS, even if the application developers use any proprietary extensions beyond SQL.

The GOSIP related issues were apparently not well founded. There were no improprieties with respect to the demonstrations that needed to be performed by the apparent winner of the bid. The protests were denied on all counts.

This decision was handed down on 9 June, 1995. An appeal could have been entered within the next 120 days, by October 9th. No one appealed this decision.

That was the procurement, the bid protests, and the subsequent decision. Unfortunately, this entire incident degenerated into religious debates about "open systems," fueled by marketing rhetoric. As stated earlier, I am very concerned that this poor reporting and "open systems" rhetoric will damage the reputation of a set of useful portability standards and specifications.

Observations and Opinions

As this topic seems so prone to "religious opinions," and indeed I may also be as blinded by my own beliefs as anyone else, I remind readers that all the following opinions and observations are my own and should not be construed as representing anything other than that.

First, I dislike the term "open systems." It may be a useful term to describe very broad concepts involving source-code portability and application interoperability, but there its usefulness ends. Source-code portability is considered one of the cornerstones of "open systems," and so the POSIX family of standards is both blessed and burdened by its association with this general and sometimes ambiguous term.

Presumably developers want to support portable software because more than a single computing environment is involved, or because they are investing in the application's future, knowing that platforms will change. Some "open systems" proponents talk about source-code portability as if it is a perfect software development exercise, where all one has to do is purchase FIPS 151-2 certified, or XPG4 Base or UNIX 95 systems, and portable applications will "happen." Even amongst traditional "UNIX" systems, source-code portability is a difficult and diverse software development problem. It is quite possible to write perfectly un-portable code on systems that support ISO C and POSIX.1. If source-code portability was easy and could be solved with a specification alone, it would have been solved with the */usr/group* 1984 standard.

The POSIX.1 specification only represents *part* of the solution. First, other POSIX projects add further functionality to this portability model. POSIX.1 was never meant to be the complete solution. No specification, however, no matter how broad, can define everything that every application is going to need.

Second, the actual building of portable applications source-code requires genuine understanding and real effort. Developers may need to develop parts of an application in a very platform specific manner for a variety of reasons, even on systems certified and branded to conform to broad, robust specifications. It is unfortunate that POSIX.0 (the POSIX Open Systems Environment Guide) and its various brethren (e.g., the NIST APP) have never described the amount of careful engineering that is required to actually implement a software development environment to support the development and maintenance of source-code portable applications, beyond the relatively simple task of itemizing standards in a profile. Source-code portability is an intensive application development concern.

Vendors have a vested interest in the fact that people moving from historical single-vendor environments to multi-vendor "open systems" environments often do not appreciate that portability is a difficult problem, requiring more than a specification on a programmer's bookshelf, and a machine with a certificate. A vendor will pay a lot of lip service to your portability and interoperability needs – providing you're buying their solution. And in their eyes, they're giving the customer exactly what they asked for. It is the customer's problem if they don't know how to use it.

POSIX.1 *is* amazingly successful. As a specification, it has existed and been stable since 1988. It has been incorporated and adopted into the most important portability specifications around us (SVID3, 4.4BSD, X/Open's XPG4 Base and its successor, the Single UNIX Specification). It has been widely implemented on systems as diverse as VMS, MVS, and MPE/iX, as well as most traditional "UNIX" systems. Most of these implementations are either certified by NIST against the FIPS 151-2 test suite, or warranted by an X/Open XPG4 brand. The predecessor to FIPS 151-2, FIPS 151-1, has existed almost from the time POSIX.1 was ratified in 1988.

Which brings us back to a particular thorny implementor. Microsoft did indeed build an exact POSIX.1 implementation. Wherever the standard has allowed them to leave something unimplemented, they did so. These loopholes exist in the standard (and therefore in its FIPS) precisely because all of the "UNIX" vendors in the working and ballot groups allowed them to exist. All of these vendors were ensuring *their* proprietary implementations of "UNIX" didn't have to change too much to conform to the standard. The NIST certification process itself offers certain leeway in the implementation (as allowed by the POSIX.1 standard) to ensure a certain breadth of competition for federal bids. Microsoft exploited these specifications no more and no less than any other vendor. But they did provide an implementation, and it is sufficiently robust to clear a rather large and robust test suite.

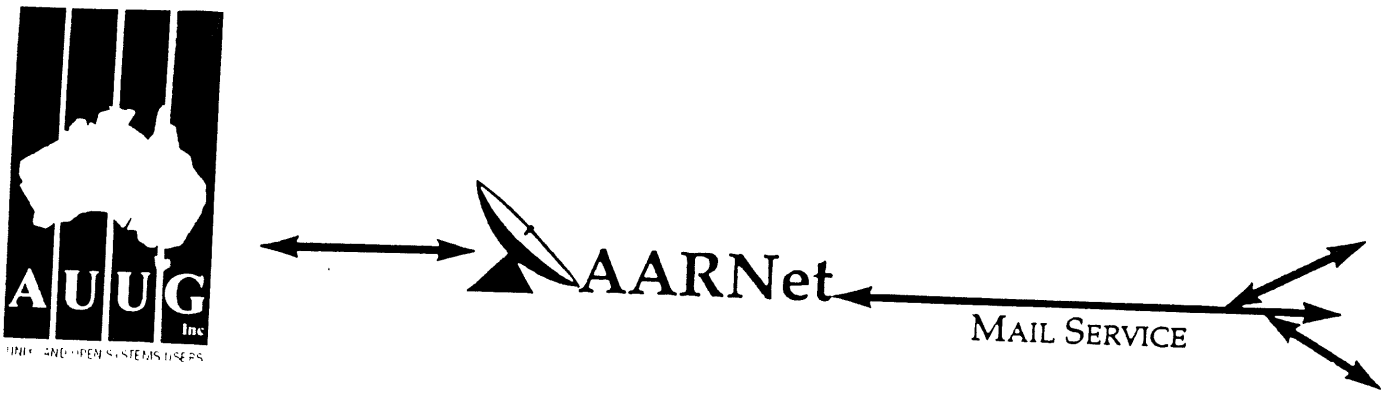
Microsoft implemented POSIX.1 as an environment subsystem, such that it is not completely integrated with the Win32 subsystem. Those that claim the integration aspects of the POSIX.1 subsystem are a problem and that the subsystem should be more integrated with Win32 should stop and think about it a little more. I care about POSIX and the C-standard because I care about portable applications. I do not want to start using OLE calls, Win32 windowing calls and such in my application. This would make it very unportable to any of the FIPS 151-2 certified or XPG4 Base branded machines that are available.

But people in our industry love to beat on Microsoft. Ten years ago, we loved to beat on IBM. In general, I think that we like to beat on whoever appears to have the most arrogant marketing campaigns, which presume that their solution is the only solution. I cannot think of a single vendor's sales team with whom I have had much patience.

Microsoft treats standards much the same way this decade as IBM did through the eighties. Standards are something they will support by implementation to satisfy customer requirements, i.e., to compete in the market, but don't necessarily support philosophically, since they most likely believe they already offer better solutions. The traditional "UNIX" vendors, however, don't appear to be doing anything differently.

Many vendors of traditional "UNIX" operating systems point to their "support for open systems standards," and scoff at Microsoft. One might ask them why they haven't yet branded their implementations as XPG4 UNIX 95? The X/Open Single UNIX Specification has existed for a year now, and its branding program has existed since March, 1995. As a superset of POSIX.1 and ISO C, XPG4 Base 95 branded systems arguably offer a better base upon which to develop new portable applications, and XPG4 UNIX 95 branded systems offer the extra bits to support porting historical "UNIX" applications to these platforms. All of the traditional "UNIX" vendors loudly proclaim their support for this effort, yet as of this writing not a single vendor has claimed the XPG4 UNIX 95 brand. Most cling to their XPG4 Base (not even Base 95) and UNIX 93 brands, so marketing rhetoric and mud-slinging comes off sounding a little shallow from my point of view.

Regardless of the vagueness and overuse of the term "open systems," developers are determining how best to use the standards and specifications that are particular to their needs, and buying platforms that provide implementations of the same and suit those requirements. This is what portability specifications are all about – providing the systems analyst or applications developer the ability to choose platforms suited to their needs (price, performance, networking, etc.) while still maintaining as much portability as possible for the application source-code developed to a particular specifications model. Whether or not the platform is a "UNIX" machine, NT, VMS, MVS, or any other system doesn't matter. It's whether it supports the relevant specifications (X/Open, POSIX, NIST FIPS) required by the application that matters.



Dear Site Administrator,

As you may be aware, the arrangements for mailing to addresses outside Australia (and also to AARNet sites) changed in May 1991. Since then, the University of Melbourne are no longer managing the administrative details associated with maintaining this service. The AARNet (Australian Academic and Research Network) management has taken over administering the service, and are requiring all ACSnet and similar sites to register with AARNet and pay a fee for continued access to Internet mail services. AARNet have set this fee as $1000 per annum for most sites, with larger sites paying more (you know who you are).

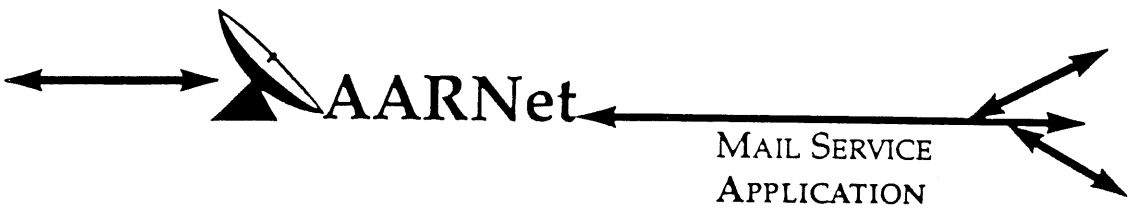
The fee is intended to cover use of AARNet bandwidth for your network traffic. Registration with AARNet, however, provides ONLY the registration of your address in worldwide address tables - your site will be unreachable without this registration. The fee does NOT cover the costs involved in obtaining a connection to AARNet or ACSnet NOR does it include a guarantee that you can be connected or even to help you find a connection point. See Note B for some information about connection services.

AUUG as a service to its members has negotiated with AARNet to achieve a lower price for this basic address registration service. The lower price is based on the reduction in paperwork for the AARNet management authorities. The AUUG/AARNet fee is dependent on the membership status of the owner of the machine(s)/domain involved, and is currently $250 for members and $600 for non-members. As such it is a substantial discount on the AARNet fee, but only applies to sites in the AARNet $1000 category. Larger sites will need to negotiate directly with AARNet.

The address registration is for one AUUG membership year. Membership years start on the 1st January or July, whichever is nearest to receipt of your application. Sites which do not renew their AUUG/AARNet registration annually with their AUUG membership each year will be removed from the Internet tables and will no longer be able to communicate with international and AARNet hosts. Reminders/invoices will be sent along with your membership renewal.

The required initial registration form is attached below. It should be completed and forwarded to AUUG's (postal) mailing address at the bottom of the form or faxed to (02) 332 4066. If you have any queries on the AUUG/AARNet arrangements please direct them to Catrina Dwyer at the AUUG office on (02) 959 3656 (catrina@swift.sw.oz.au) or myself (frank@atom.ansto.gov.au).

Regards,
Frank Crawford
AUUG-AARNET Administrator
AUUG Inc.



MAIL SERVICE APPLICATION

On behalf of the organisation listed below I wish to apply to be a Mail Service Affiliate Member of AARNet, and accordingly request that AUUG Incorporated arrange for the Australian Vice-Chancellors' Committee (AVCC) to maintain on my behalf an electronic mail delivery record in the Australian Academic and Research Network (AARNet) to allow my organisation to send and receive electronic mail carried across AARNet.

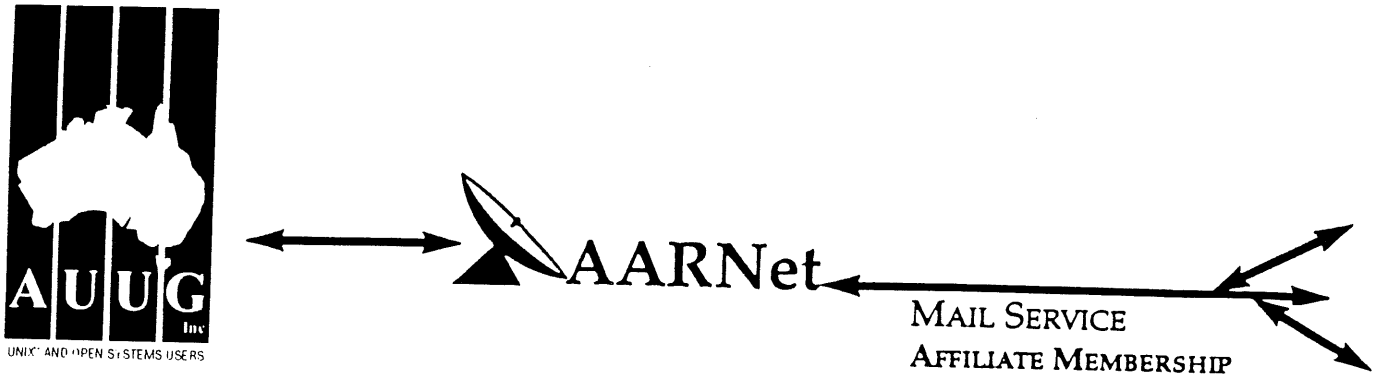
I understand that the AVCC may consult the recorded logs of my organisation's usage of AARNet facilities for 1990, and determine that I am ineligible for registration under the terms of the agreement between AVCC and AUUG Inc. I understand that AUUG Inc will invoice my organisation for this service for the calendar year 1991 and for subsequent years unless it receives my organisation's written advice to terminate the Affiliate Membership of AARNet.

I understand that the AVCC and AUUG Inc maintain the right to vary the Mail Service Affiliate Membership charges from year to year, and maintains the right to cease offering this service to my organisation at the start of any year, at their discretion. I understand that in the event of any variation of the Mail Service Affiliate Membership of AARNet, my organisation will be advised in writing by the AVCC or AUUG Inc to the address below.

I understand that in consideration of the AARNet Mail Service Affiliate Membership charge, AARNet will undertake to maintain a mail directory entry which will direct incoming electronic mail to the AARNet gateway system(s) which I have nominated below. Furthermore I accept that there is no other undertaking made by AARNet in terms of reliability of mail delivery or any other form of undertaking by AARNet or the AVCC in consideration of the payment to AARNet for the maintenance of the mail directory entry on AARNet.

I undertake that my organisation's use of the mail delivery services over AARNet will not be used as a common commercial carrier service between my organisation and other organisations receiving similar services from AARNet, nor will it be used as a commercial carrier service between branches of my organisation. Furthermore my organisation undertakes to use AARNet facilities within the terms and conditions stated in the AARNet Acceptable Use Policy. I accept the right of the AVCC or AUUG Inc to immediately terminate this service at their discretion if these undertakings are abused by my organisation (where the AVCC retains the right to determine what constitutes such abuse).

I understand that a fee is payable with this application: of $250 if the host/hosts covered are owned by a member of AUUG Incorporated, or $600 if the host/hosts covered are not owned by an AUUG member. Corporation host owners may only claim the member price if the corporation is an Institutional member of AUUG Inc. My cheque payment of either $250 or $600 as appropriate is enclosed with this application.



MAIL SERVICE
AFFILIATE MEMBERSHIP
APPLICATION FORM cont'd.

Note A. Mail Delivery Information

Two items of information are required: firstly the preferred name of your mail host (or the domain name(s) of a group of hosts) in Internet domain name system format, and secondly the name (or names) or AARNet gateway systems who will accept electronic mail over AARNet (and connected overseas networks) on your behalf and forward it to you. The primary requirement for an AARNet gateway is its ability to recognise your host/domain addresses and perform the necessary mail header rewriting reliably.

Please check with the postmaster at your preferred AARNet gateway host site before citing them as a gateway for AARNet mail delivery. For ACSnet addresses (*.oz.au), the host "munnari.oz.au" (Melbourne University) is a recommended gateway. Other possible sites include "metro.ucc.su.oz.au" (Sydney University), sirius.ucs.adelaide.edu.au (University of Adelaide), uniwa.uwa.oz.au (University of WA) and bunyip.cc.uq.oz.au (University of Qld). Note that all gateway addresses must be fully domain qualified.

Example Mail Directory Information request:

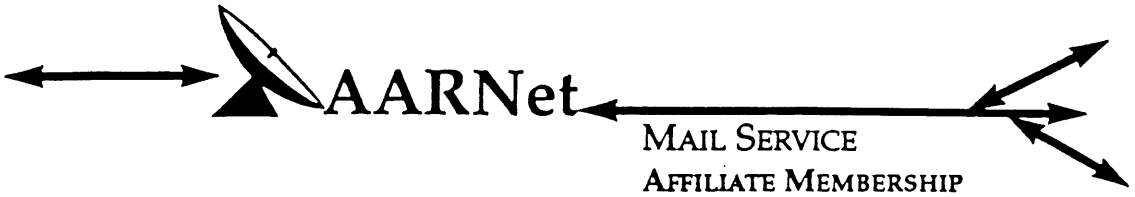
Mail addresses required:	acme.oz.au, *.acme.oz.au
Mail Gateways (primary)	gw.somewhere.edu.au
(secondary)	munnari.oz.au
(secondary)	unnet.uu.net

The addressability of your site and the willingness of your nominated gateways to act in that capacity will be determined before registration proceeds. Processing will be made faster if you contact the postmaster at your nominated gateways in advance to inform them of your intentions. Your nominated technical contact will be notified by email when registration is complete.

Note B. Getting Connected

New sites will need to find an existing AARNet or ACSnet site who will accept their site as a connection, and also select a protocol for transferring data over their mutual link. Although the UUCP package is a standard inclusion with UNIX, it is little used in Australia due to its relatively poor performance. Other possible choices for your link protocol include SLIP (TCP/IP) and MHSnet.

Among a number of organisations who provide connection services, Message Handling Systems Pty Ltd have announced a special offer on both their link software and connect time for AUUG members. For more details on this offer, contact Message Handling Systems on (02) 550 4448 or elaine.mhs.oz.au.



MAIL SERVICE
AFFILIATE MEMBERSHIP
APPLICATION FORM

PLEASE PRINT CLEARLY!

Date:_____

Name of Organisation/Owner:_____

_____

Signed:_____AUUG Membership No (if known):_____

Name:_____Position:_____

on behalf of the organisation named above.

Address:_____

_____

_____

_____Postcode:_____

Administrative Contact:_____ Title:_____

E-Mail:_____ Phone: (___) _____

Fax: (___) _____

Technical Contact:_____ Title:_____

E-Mail:_____ Phone: (___) _____

Fax: (___) _____

Mail Delivery Information to be entered in AARNet (see Note A next page)

Domain Names Requested: _____

_____

Gateway Addresses:_____

_____

_____

Expected Link Protocol: UUCP SL/IP MHSnet Other:_____

* * * *

Send this page only to:

AUUG Incorporated
PO Box 366
Kensington NSW 2033

Phone: +61 2 361 5994
Fax: +61 2 332 4066

MEMBERSHIP APPLICATION
INDIVIDUAL

AUUG Inc
UNIX® AND OPEN SYSTEMS USERS

To apply for AUUG membership, complete this form and return it with payment in Australian Dollars to:

REPLY PAID 66, AUUG MEMBERSHIP SECRETARY,
P.O. BOX 366, KENSINGTON, NSW 2033, AUSTRALIA
Tel: +61 2 361-5994 or 1 800 625 655 • Fax: +61 2 332-4066

Tick this box if you wish your name withheld from mailing lists made available to vendors. ☐

NOTE: Please do not send purchase orders - perhaps your purchasing department will consider this form to be an invoice. Foreign applicants please send a bank draft drawn on an Australian bank.

INDIVIDUAL OR STUDENT MEMBERSHIP:

I, _____ do hereby apply for:

Renewal/New membership of AUUG	☐ $ 90.00
Renewal/New Student membership	☐ $ 25.00 (please complete certification portion)
International air mail	☐ $ 60.00

I agree that this membership will be subject to the rules and by-laws of AUUG as in force from time to time, and that this membership will run from time of joining/renewal until the end of the calendar or financial year.

_____
Signature

TOTAL REMITTED: AUD$_____ (Cheque, or money order, or credit card)

_____
Date

LOCAL CHAPTER DESIGNATE:

You can participate in the activities of a local AUUG Chapter. Part of your fee will be given to the chapter to support those activities. By default a regional chapter will be selected for you. If you would rather nominate a chapter, please specify here _____ (indicate NONE for no chapter).

TO BETTER SERVE YOU, PLEASE PRINT YOUR CONTACT INFORMATION:

Name/Contact: _____
Position/Title: _____
Company: _____
Address: _____
_____ Postcode _____
Tel: BH _____ AH _____
Fax: BH _____ AH _____
email address: _____

Over for Institutional Membership

STUDENT MEMBER CERTIFICATION: *(to be completed by a member of the academic staff)*

I, _____ certify that
(administrator)
_____ is a full time student at
(name)
_____ and is expected to
(institution)
graduate approximately _____.
(date)

_____
Signature

_____ _____
Title *Date*

Please charge $_____ to my
☐ Bankcard, ☐ Visa, ☐ Mastercard
Account number: _____
Expiry date: _____
Name on card: _____
Signature: _____

AUUG Secretariat Use

Chq: bank _____ *bsb* _____
a/c _____ *#* _____
Date: _____ *$* _____
Initial: _____
Date processed: _____
Membership # _____

AUUG Inc. as a user group, exists to provide UNIX® and open systems users with relevant and practical information, services, and education through cooperation among users.

MEMBERSHIP APPLICATION
INSTITUTIONAL

UNIX® AND OPEN SYSTEMS USERS

To apply for AUUG membership, complete this form and return it with payment in Australian Dollars to:

REPLY PAID 66, AUUG MEMBERSHIP SECRETARY,
P.O. BOX 366, KENSINGTON, NSW 2033, AUSTRALIA
Tel: +61 2 361-5994 or 1 800 625 655 • Fax: +61 2 332-4066

Tick this box if you wish your name withheld from mailing lists made available to vendors. ☐

NOTE: Please do not send purchase orders - perhaps your purchasing department will consider this form to be an invoice. Foreign applicants please send a bank draft drawn on an Australian bank.

We, _____
(Company Name)

do hereby apply for:

Renewal/New* Inst. membership of AUUG ☐ $350.00
International air mail ☐ $120.00

TOTAL REMITTED AUD$_____
(Cheque, money order, or credit card)

I/We agree that this membership will be subject to the rules and by-laws of AUUG as in force from time to time, and that this membership will run from time of joining/renewal until the end of the calendar or financial year.

I/We understand that I/we will receive two copies of the AUUG newsletter, and may send two representatives to AUUG sponsored events at member rates, though I/we will have only one vote in AUUG elections, and other ballots as required.

Signed_____ Date _____

Title _____

INSTITUTIONAL MEMBER DETAILS:
To be completed by institutional members only.

Following are our specified contacts. The primary contact holds the full member voting rights. The two designated reps will also be given membership rates to AUUG activities including chapter activities. By default a regional chapter will be selected for you. If you would rather nominate a chapter, please specify in space provided (indicate NONE for no chapter). *(Please print clearly or type)*

Primary Contact_____

Position/Title _____

Address _____

_____ Postcode_____

Bus. Tel: _____ Bus. Fax:_____

e-mail address _____

Local Chapter Pref. _____

1st Rep._____

Position/Title _____

Address _____

_____

Bus. Tel: _____ Bus. Fax:_____

e-mail Address _____

Local Chapter Pref. _____

2nd Rep._____

Position/Title _____

Address _____

_____

Bus. Tel: _____ Bus. Fax:_____

e-mail address _____

Local Chapter Pref. _____

Please charge $_____ to my
☐ Bankcard, ☐ Visa, ☐ Mastercard

Account number:_____

Expiry date:_____

Name on card:_____

Signature:_____

AUUG Secretariat Use

*Chq: bank*_____ *bsb* _____
a/c _____ *#* _____
Date: _____ *$* _____
Initial: _____
Date processed: _____
Membership # _____

AUUG Inc. as a user group, exists to provide UNIX® and open systems users with relevant and practical information, services, and education through cooperation among users.

Notification of Change

AUUG Inc
UNIX® AND OPEN SYSTEMS USERS

You can help us! If you have changed your mailing address, phone, title, or any other contact information, please keep us updated. Complete the following information and either fax it to the AUUG Membership Secretary on (02) 332-4066 or post it to:

AUUG Membership Secretary
P.O. Box 366
Kensington, NSW 2033
Australia

(Please allow at least 4 weeks for the change of address to take effect..)

❒ The following changes are for my personal details, member #:_____

❒ The following changes are for our Institutional Member, primary contact.

❒ The following changes are for our Institutional Member, representative 1.

❒ The following changes are for our Institutional Member, representative 2.

PLEASE PRINT YOUR OLD CONTACT INFORMATION (OR ATTACH A MAILING LABEL):	PLEASE PRINT YOUR NEW CONTACT INFORMATION:
Name/Contact: _____	Name/Contact: _____
Position/Title: _____	Position/Title: _____
Company: _____	Company: _____
Address: _____	Address: _____
_____ Postcode _____	_____ Postcode _____
Tel: BH _____ AH _____	Tel: BH _____ AH _____
Fax: BH _____ AH _____	Fax: BH _____ AH _____
email address: _____	email address: _____

AUUG Secretariat Use

Date: _____
Initial: _____
Date processed: _____
Membership # _____