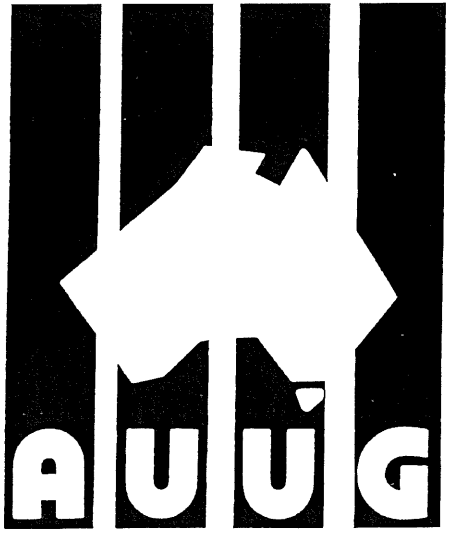


ISSN 1035-7521

AUUGN

AUUG Inc. Newsletter

Volume 14, Number 3

June 1993

Registered by Australia Post, Publication Number NBG6524

W Teeuwen

The AUUG Incorporated Newsletter

Volume 14 Number 3

June 1993

CONTENTS

AUUGN is the journal of AUUG Incorporated, an organisation with the aim of promoting knowledge and understanding of Open Systems including but not restricted to the UNIX* system, networking, graphics, user interfaces and programming and development environments, and related standards.

* UNIX is a registered trademark of UNIX System Laboratories, Incorporated

AUUG General Information

Memberships and Subscriptions

Membership, Change of Address, and Subscription forms can be found at the end of this issue.

Membership and General Correspondence

All correspondence for the AUUG should be addressed to:-

The AUUG Secretary,
P.O. Box 366,
Kensington, N.S.W. 2033.
AUSTRALIA

Phone: (02) 361 5994
Fax: (02) 332 4066
Email: auug@munnari.oz.au

AUUG Business Manager

Liz Fraumann,
P.O. Box 366,
Kensington, N.S.W. 2033.
AUSTRALIA

Phone: +61 2 953 3542
Fax: +61 2 953 3542
Email: eaf@softway.sw.oz.au

AUUG Executive

President

Phil McCrea
phil@softway.oz.au
Softway Pty. Ltd.
79 Myrtle Street
Chippendale NSW 2008

Vice-President

Glenn Huxtable
glenn@cs.uwa.oz.au
University of Western Australia
Computer Science Department
Nedlands WA 6009

Secretary

Peter Wishart
pjw@lobo.canberra.edu.au
EASAMS Australia
Level 6
60 Marcus Clark St.
Canberra ACT 2600

Treasurer

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

Committee Members

Rolf Jester
rolf.jester@sno.mts.dec.com
Digital Equipment Corporation
P O Box 384
Concord West NSW 2138

Chris Maltby
chris@softway.sw.oz.au
Softway Pty. Ltd.
79 Myrtle Street
Chippendale NSW 2008

John O'Brien
john@wsa.oz.au
Whitesmiths Australia P/L
#5 Woods Centre
ANSTO Business & Tech. Park
Lucas Heights NSW 2234

Michael Paddon
mwp@iconix.oz.au
Iconix Pty Ltd
851 Dandenong Rd
East Malvern VIC 3145

Greg Rose
ggr@acci.com.au
ACCI
723 Swanston St
Carlton VIC 3053

AUUG General Information

Next AUUG Meeting

The AUUG'93 Conference and Exhibition will be held from the 27th to 30th September, 1993, at the Sydney Convention and Exhibition Centre, Darling Harbour, Sydney.

Advertising

Advertisements to be included in AUUGN are welcome. They should conform to the standards of other contributions (see page 5). Advertising rates are $120 for a quarter page, $180 for half a page, $300 for the first A4 page, $250 for a second page, $500 for the inside cover and $750 for the back cover. There is a 20% discount for bulk ordering (ie, when you pay for three issues or more in advance). Contact the business manager for details.

Mailing Lists

For the purchase of the AUUGN mailing list, please contact the AUUG secretariat, phone (02) 361 5994, fax (02) 332 4066.

Back Issues

Various back issues of the AUUGN are available. For availability and prices 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 the Conference Proceedings for AUUG'92 are still available, at $50 each. Contact the AUUG secretariat.

Acknowledgement

This newsletter was produced with the kind assistance of and on equipment provided by the Australian Nuclear Science and Technology Organisation.

Disclaimer

Opinions expressed by authors and reviewers are not necessarily those of AUUG Incorporated, its Newsletter or its editorial committee.

AUUG Newsletter

Editorial

Welcome to AUUGN Volume 14 Number 3. In this issue we have a number of summaries on conferences attended by our people, and others run by us. We have Liz's views on the UniForum conference, Phil reports on the Fiji Society Conference and overviews on the Perth and Melbourne summer conferences. Three papers, one each from the Sydney, Perth and Queensland conferences, have also been included.

On the chapter side, we have accepted four new chapters (see Peter's article) and have reports from both Perth and Melbourne on their chapter activities.

Other articles include, information on SAGE-AU, the findings of the survey carried out by DATAPRO on Australian Users' Views on Open Systems, and AARNet information. Also, a number of book reviews have been provided for this issue of AUUGN, with more coming in the future.

Last, but not least, I am looking at expanding both the types of things published in AUUGN and the people involved in its preparation. For more details see the 'Help Wanted' on page 8, but remember that if no one responds some of these ideas cannot be done.

Jagoda Crawford

AUUGN Correspondence

All correspondence regarding the AUUGN should be addressed to:-

AUUGN Editor,
P.O. Box 366,
Kensington, N.S.W. 2033.
AUSTRALIA

Phone: +61 2 717 3885
Fax: +61 2 717 9273
Email: auugn@munnari.oz.au

AUUGN Book Reviews

The AUUGN book review editor is Frank Crawford. Anyone interested in reviewing books or with book reviews to submit for publishing in AUUGN please contact Frank. His address can be found on page two of this issue. Remember, that any books you review, you keep.

Contributions

The Newsletter is published approximately every two months. The deadlines for contributions for the next issues of AUUGN are:

Volume 14 No 4	Friday 30th July
Volume 14 No 5	Friday 24th September
Volume 14 No 6	Friday 26th November

Contributions should be sent to the Editor at the above address.

I prefer documents to be e-mailed to me, and formatted with troff. I can process mm, me, ms and even man macros, and have tbl, eqn, pic and grap preprocessors, but please note on your submission which macros and preprocessors you are using. If you can't use troff, then just plain text or postscript please.

Hardcopy submissions should be on A4 with 30 mm margins, and 30 mm left at the bottom so that the AUUGN footers can be pasted on to the page. Small page numbers printed in the footer area would help.

AUUG Institutional Members as at 04/06/1993

A. Goninan & Co. Limited
A.N.U.
AAII
Adept Software
Alcatel Australia
Allaw Technologies
Amalgamated Television Services
Amdahl Pacific Services
Andersen Consulting
ANI Manufacturing Group
ANSTO
Anti-Cancer Council of Victoria
ANZ Banking Group/I.T. Development
Attorney Generals' Dept
Attorney-General's Dept
AUSOM Inc.
Ausonics Pty Ltd
Auspex Systems Australia
Australian Airlines Limited
Australian Archives
Australian Bureau of Agricultural and Resource Economics
Australian Bureau of Statistics
Australian Computing & Communications Institute
Australian Defence Industries Ltd
Australian Electoral Commission
Australian Museum
Australian National Parks & Wildlife Service
Australian Software Innovations
Australian Taxation Office
Australian Technology Resources
Australian Technology Resources (ACT) Pty Ltd
Australian Wool Corporation
Automold Plastics Pty Ltd
AWA Defence Industries
B & D Australia
Bain & Company
BHA Computer Pty Limited
BHP CPD Research & Technology Centre
BHP Information Technology
BHP Minerals
BHP Petroleum
BHP Research - Melbourne Laboratories
BHP Research - Newcastle Laboratories
BICC Communications
Bond University
Burdett, Buckeridge & Young Ltd.
Bureau of Meteorology
Bytecraft Pty Ltd
C.I.S.R.A.
C.I.S.U.
Cadcom Solutions Pty. Ltd.
Cape Grim B.A.P.S
Capricorn Coal Management Pty Ltd
CelsiusTech Australia
Chief Secretary's Dept
CITEC
Classified Computers Pty Ltd
Co-Cam Computer Group
Codex Software Development Pty. Ltd.
Cognos Pty Ltd
Colonial Mutual
Com Net Solutions
Com Tech Communications
Commercial Dynamics
Communica Software Consultants
Composite Buyers Ltd
Computechnics Pty Ltd
Computer De Tokyo Corporation
Computer Sciences of Australia Pty Ltd
Computer Software Packages
Computer Systems (Australia) Pty. Ltd.
Corinthian Engineering Pty Ltd
Corporate Workgroup Resources
CSIRO
Curtin University of Technology
Customised Software Solutions Centre
Cyberdyne Systems Corporation Pty Ltd
Cyberscience Corporation Pty Ltd
Data General Australia
Datacraft Technologies
Deakin University
Deakin University
Defence Housing Authority
Defence Service Homes
Department of Family Services & Aboriginal & Islander Affairs
Dept of Agricultural & Rural Affairs
Dept of Business & Employment
Dept of Defence
Dept of Education, Qld
Dept of Industrial Relations, Employment, Training & Further Education
Dept of Planning & Housing
Dept of the Premier and Cabinet
Dept. of Conservation & Environment
Dept. of Defence
Dept. of the Premier and Cabinet
Dept. of the Treasury
Dept. of Transport
DEVETIR
Digital Equipment Corp (Australia) Pty Ltd
Easams (Australia) Ltd
EDS (Australia) Pty Ltd
Electronic Financial Services Limited
Emulex Australia Pty Ltd
Equinet Pty Ltd
Ericsson Australia Pty Ltd
ESRI Australia Pty Ltd
FGH Decision Support Systems Pty Ltd
Financial Network Services
Fire Fighting Enterprises
First State Computing
Flinders University
Fremantle Port Authority
Fujitsu Australia Ltd
G. James Australia Pty Ltd
GCS Pty Ltd
Geelong and District Water Board
Genasys II Pty Ltd
General Automation Pty Ltd
GeoVision Australia
GIO Australia
Golden Circle Australia

AUUG Institutional Members as at 04/06/1993

Great Barrier Reef Marine Park Authority
Gribbles Pathology
Gunnedah Abattoir
Haltek Pty Ltd
Hamersley Iron
Harris & Sutherland Pty Ltd
Hermes Precisa Australia Pty. Ltd.
Honeywell Ltd
Honeywell Ltd
Hong Kong Jockey Club Systems (Australia) Pty Ltd
I.B.A.
IBM Australia Ltd
Iconix Pty Ltd
Information Technology Consultants
Insession Pty Ltd
Insurance & Superannuation Commission
International Imaging Systems
Internode Systems Pty Ltd
Ipec Management Services
IPS Radio & Space Services
James Cook University of North Queensland
JTEC Pty Ltd
Knowledge Engineering Pty Ltd
KPMG Solutions
Labtam Australia Pty Ltd
Land Titles Office
Leeds & Northrup Australia Pty. Limited
Logica Pty Ltd
Logical Solutions
Macquarie University
Mayne Nickless Courier Systems
McDonnell Douglas Information Systems Pty Ltd
Medical Benefits Funds of Australia Ltd.
Mentor Technologies Pty Ltd
Meridian Information Services Pty Ltd
Metal Trades Industry Association
Mincom Pty Ltd
Minenco Pty Ltd
Mitsubishi Motors Australia Ltd
Mitsui Computer Limited
Moldflow Pty. Ltd.
Motorola Computer Systems
Multibase Pty Ltd
National Library of Australia
NCR Australia
NEC Australia Pty Ltd
Northern Territory Library Service
Northern Territory University
NSW Agriculture
Ochre Development
Office of Fair Trading
Office of National Assessments
Office of the Director of Public Prosecutions
Olivetti Australia Pty Ltd
Open Software Associates Ltd
Oracle Systems Australia Pty Ltd
OSIX Pty Ltd
OzWare Developments Pty Ltd
Pacific Star Communications
Paxus
Philips PTS
Port of Melbourne Authority
Powerhouse Museum
Prentice Hall Australia
Process Software Solutions Pty Ltd
Prospect Electricity
pTizan Computer Services Pty Ltd
Public Works Department
Pulse Club Computers Pty Ltd
Pyramid Technology Corporation Pty Ltd
Qantek
Quality By Design Pty Ltd
Redland Shire Council
Release4
Renison Golfields Consolidated Ltd
Rinbina Pty Ltd
Royal Melbourne Institute of Technology
SBC Dominguez Barry
Scitec Communication Systems
Sculptor 4GL+SQL
SEQEB Business Systems
SEQEB Control Centre
Shire of Eltham
Siemens Nixdorf Information Systems Pty Ltd
Snowy Mountains Authority
Software Developments
Softway Pty Ltd
Sony Technology Centre of Australia
South Australian Lands Dept
St Vincent's Private Hospital
St. Gregory's Armenian School
Stallion Technologies Pty Ltd
Standards Australia
State Bank of NSW
State Super (SSIMC)
Steedman Science and Engineering
Steelmark Eagle & Globe
Swinburne Institute of Technology
Sydney Electricity
Sydney Ports Authority
System Builder Development Pty Ltd
Systems Development Telecom Australia
TAB of Queensland
Tandem Computers
Tattersall Sweep Consultation
Technical Software Services
Telecom Australia
Telecom Australia Corporate Customer
Telecom Network Engineering Computer Support Services
Telecom Payphone Services
The Far North Qld Electricity Board
The Fulcrum Consulting Group
The Opus Group Australia Pty Ltd
The Preston Group
The Roads and Traffic Authority
The Southport School
The University of Western Australia
Thomas Cook Ltd.
TNT Australia Information Technology
Toshiba International Corporation Pty Ltd
Tower Software Engineering Pty Ltd

AUUG Institutional Members as at 04/06/1993

Tower Technology Pty Ltd
Tradelink Plumbing Supplies Centres
Triad Software Pty Ltd
Turbosoft Pty Ltd
TUSC Computer Systems
UCCQ
Unidata Australia
Unisys
Unisys Australia Ltd
UNIVEL
University of Adelaide
University of Melbourne
University of New South Wales
University of Queensland
University of South Australia
University of Sydney
University of Tasmania
University of Technology, Sydney
UNIX System Laboratories
Unixpac Pty Ltd
Vicomp
Victoria University of Technology
VME Systems Pty Ltd
Wacher Pty Ltd
Walter & Eliza Hall Institute
Wang Australia Pty. Ltd.
Water Board
Western Mining Corporation
Work Health Authority
Workstations Plus
Zircon Systems Pty Ltd
Zurich Australian Insurance

Help Wanted

The following are some new sections that I would like to include in future issues of AUUGN. I would like to hear from members out there that would like to assist with any of the following. Also any other ideas that you might have for changes or additions are welcomed.

- Local Chapter Activities - at this stage some of the chapters are supplying information on their activities. It would be useful for a representative from each chapter to gather the information and submit it to AUUGN on a regular basis. Further, someone to co-ordinate all these reports would also help (otherwise I'll look after it myself).
- Conference Summaries - not all AUUG members can afford the time or money to attend conferences around the world. Further, some people occasionally represent AUUG at such conferences (almost always at their own or the host groups' expense). What we would like is reports from members about the conferences they attend (whether privately or not), so that others can hear of what is going on in the outside world. This also includes reports on both, the AUUG Winter conference and the Summer conference series, as not all members can attend them either.
- AUUGN has a long history, it has been published for quite a number of years now, and we are regularly getting queries, asking if we have ever published this or that. We are looking for someone to create a complete index of all the AUUGN issues, from volume one to now. This will probably be in *refer* format and probably included in the USENIX project at a latter stage.
- AUUG receives a number of Press Releases. A condensed summary of these should be published in AUUGN. Anyone interested in doing this (preferably someone with a marketing/commercial background)?
- We are currently re-printing the User Support Corner from YAUN (WAUG's newsletter). We should have one of our own. Any takers?
- Any other ideas?

Jagoda Crawford
jc@atom.ansto.gov.au

AUUG President's Report

Four Letter Acronyms (FLA) are taking over ...

The computer and telecommunications industries are characterized by Three Letter Acronyms (TLAs) – from company names such as IBM and DEC, to protocols such as SNA etc. Organizations such as Telecom are perhaps the most creative (ie prolific) when it comes to TLAs.

The onset of open systems (read UNIX) has ushered in the era of Four Letter Acronyms (FLAs)! Have you picked up a journal recently that has not had features on WABI and COSE?? And what do these terms mean?

COSE

The Common Open Software Environment (COSE) is a belated attempt to get some standardization at the applications level, to ensure ease of portability of application software. Machine manufacturers by and large have been between a rock and a hard place in the past few years, touting themselves to be 'open', but at the same time looking for product differentiation which will set them apart from the rest. One wag I know says that the only common ground amongst machine suppliers regarding *open* systems is that it is used to *open* people's wallets!

And so we have had a succession of UNIX machine providers all wishing to enlarge their market share, but offering machines with a 'standards' compliant operating system which happens to lock users in to that operating system, perhaps because of the chipset being used (eg SPARC), or because of some nice add-on features to the operating system.

Whilst the major UNIX suppliers were all at war with each other arguing about which one was the most 'open', a well known manufacturer of PC operating systems was announcing a new operating system, which would be the logical successor to DOS. Suddenly there emerged unity in the face of adversity! COSE was born overnight. Its scope isn't just the operating system – it's the entire desktop. Sun in particular has finally capitulated in relation to the user interface, by giving the nod to Motif under COSE.

How fickle alliances are! The old OSF triumvirate of IBM/HP/DEC has been split in relation to COSE, with IBM and HP being part of COSE, and DEC (apparently) not being asked to play ball. Are the other players suspicious of the cosy (no pun intended) relationship between DEC and Microsoft? The speculators, however, would have us believe that DEC will soon be part of the COSE deal.

WABI

This new Applications Binary Interface (ABI) is designed to let Windows programs run on UNIX machines. Like COSE, it is long overdue, and has been spearheaded by Sun Microsystems. Sun, like several other vendors, has realized that it is not the operating system which users care about – it is the snappy DOS applications which users like.

The average user doesn't care whether a product they are using conforms to X/Open's Portability Guide, even if they have heard of it!! All they want to do is to be able to run Lotus 1 2 3, or whatever, on one of these snazzy new workstations, on which they will be able to do lots of other things at the same time!

WABI, which has been developed by SunSelect, one of the Sun divisions, allows Windows applications to run under UNIX without the need for MS-DOS or Windows. USL, SCO and SunSoft will each integrate WABI into their respective UNIX offerings, while others such as Toshiba, Fujitsu, and NCD have lent support. It is mooted that IBM and HP will soon join in.

WABI is a healthy development, and it is a clear recognition that the organization which provides an operating environment which will run the existing huge base of installed DOS applications without modification will win the hearts and minds (and pockets) of desktop users.

This observer is watching both COSE and WABI developments with interest...

P. McCrea

Mind-share vs. Market-share
A Perspective

By Liz Fraumann

In March I attended the UniForum conference and exhibition in San Francisco, CA, USA, on behalf of AUUG Inc., as an affiliate representative. It was a typical U.S. tradeshow with nearly 32,000 attendees viewing the newest wares and technology provided by the industry's computer giants and the hopefuls. It was the organisation's 10th anniversary. There was a gala atmosphere, a video made just for the occasion, and grand receptions.

A highlight of the conference was a "bash" hosted by AT&T at the San Francisco "Giftcentre." This was a UNIX® star studded evening with the long list of who's who, all present and accounted for. Done completely in the style of the Oscar's, names from UNIX's beginning, through today, graced the centre stage to receive a crystal award. Our own John Lions and Robert Elz were up for nominations... although like Clint Eastwood for many years in the past... this was not to be their year.

Since early on the UNIX community has been more like a clique, or a "religion" (I guess some religions do believe in daemons...). This gathering once again appeared a gathering of the converted, the believers, and that inner clique. UNIX can and does offer real business solutions to very complex problems. While the converted are well aware of that fact, it gets lost in its outward bound message to the non-believers.

Two presentations are very prominent in their affect on the audience. First, the third day keynote panel with Roel Pieper of UNIX System Laboratories, David Tory of the Open Software Foundation, and ... Paul Maritz of ... Microsoft. Second, is a report synopsis given by Dun & Bradstreet of a survey accomplished on site.

Looking closely at the keynote panel:
The topic was, *Open Systems: Executive Perspective*. Each panelist had 7 minutes to give his perspective on open systems, where it stands, and a view of the future. Roel Pieper lead the way, giving, as usual, an interesting and decent presentation. Dave Tory, using no foils or slides gave his perspective, not that different from Roel. Paul, on the other hand, proceeded to "plug-in" his notebook computer, (during Dave's talk, I might add, which was "not-cool dude") and *presto* a full "animated" presentation was at hand. Asked by the audience later, yes, it was running NT. This is **mind-share** gathering. Microsoft staff is highly skilled at this. UNIX has this capability and much more! Roel Pieper gave a "Destiny" demo last year at an AUUG conference BoF which also "wowed" the crowds... but unfortunately he did not use it here.

UNIX is a registered trademark of UNIX System Laboratories in the U.S. and other countries.

The moderator, Esther Dyson, a highly respected analyst in the U.S., then asked for the house lights to come up and the audience to participate. She asked for a show of hands...

- How many of you are Users? (We all assumed she meant of UNIX)...
  - Many hands
- How many of you are ISV's?
  - A few hands
- How many of you are vendors?
  - A medium amount of hands
- How many of you want to see more development of the systems?
  - About 5 hands out of the total audience...
- How many of you want to see more applications developed?
  - Almost the entire audience, and many raised both hands!

Esther then redirected to the panelists... and to my dismay, they continued to discuss the systems! They did not "listen" to the audience. This activity had been a quick market analysis. These were the converted UNIX clique sitting in the audience, and yet these leaders proceeded down a very narrow tunnel. I believe opportunity awaits organisations who help their customers, and potential customers, solve their solutions. UNIX offers many advantages over others operating systems in the marketplace. But to win customers, you must have the mind-share as well.

The Dun & Bradstreet survey was extremely interesting. Just outside the registration area were a bank of computers. A large sign over them read, "Your opinion counts," "Take 5 minutes to let us hear from you!" 1,200 respondents in 2 days took that 5 minutes. The outcome, as presented at the beginning of the third day with an accompanying press release shook a few people up. I submitted the press release in the last *AUUGN* but for those who did not see it, four main points were released:

1) VIEWS ON OPEN SYSTEMS:
   - What defines open systems?
     63% felt it is defined by interoperability. Note it is not defined by UNIX
   - Why are open systems important?
     64% to increase flexibility
   - Have you experience financial benefit since moving to open systems?
     Only 2% had any negative impact.

2) THE CREDIBILITY OF UNIX
   - What do you consider to be the best source of standards for UNIX?
     Split into two categories, general and media
     - General - 41% Standards defining organisations i.e. IEEE, 28% defacto
     - Media - 12% Standards organisations, 42% defacto

3) VIEWS ON RIGHTSIZING
- 77% are planning on it.
- 85% felt if have financial exposure it will provide good savings.

4) NT vs UNIX (this one blew a lot of people away)
Do you believe NT will offer significant benefits above UNIX?
- UniForum members - 37% felt it will offer more
- Exhibitors - 43% felt it will offer more
- Media - 50% felt it will offer more

Point 4 had a powerful impact. Returning to Australia and following up with Dun & Bradstreet for a copy of the full survey report has proven to be most enlighting.

The actual results look like this:

Do you believe NT will offer significant benefits above UNIX?;

- 37% yes
- 42% no
- 22% don't know/no answer

While math is not one of my strengths, I read this as 42% of the respondents did NOT feel NT will offer significant benefits above UNIX. However, what was reported in the presentation is, of the 37% who felt it would, 50% were media, 43% were exhibitors, and 37% were UniForum members. This is still a statistic that should raise awareness but not something the UNIX community should panic over. The fundamental part of open systems is the opportunity of choice.

To win the mind-share, UNIX providers must reach the unconverted and provide solutions and applications that the users need and desire. I believe the AUUG conference and exhibition this year is a natural forum for this to take place. It allows potential customers to view potential solutions. It is up to the vendors, ISV's, systems integrators, and consultants to take this opportunity and go beyond the typical tradeshow practice. They must work together with their current customers, share how UNIX is already being used in the "everyday" computing arenas of retail markets, services industries, manufacturing, etc., and allow new customers to closely identify with others in their markets.

AUUG '93's planned "AUUG Village" is where it will all happen. A quotation from a movie, *The Dead Poet Society* applies... "Seize the day!"

1993 UniForum Publication Order Form

Name _____

Title _____

Company _____

Address _____ Mail Stop _____

City/State _____ Zip _____

Country _____ Telephone (____) _____

Fax (____) _____ Best time to contact: a.m. p.m.

Member Status: ☐ Member (#_____) ☐ Non-member ☐ Please send membership information

---

METHOD OF PAYMENT ☐ Check payable to UniForum ☐ Money Order (U.S. dollars)
☐ MasterCard ☐ Visa ☐ American Express

Credit Card Number _____ Exp. Date_____/ _____

Signature _____ Print name _____

Payment must be included in U.S. dollars with all orders. International checks must be drawn on a U.S. bank. Credit card orders are also accepted by telephone. Enter the appropriate information and return with payment to UniForum.

Where did you hear about UniForum? _____
☐ Check here if you do NOT want your name on the Voluntary Mail List.

PUBLICATION ORDERS

	Category		Postage/Handling			
	Member Price	Non-Member Price	Domestic	Canada	Overseas	
UniForum Monthly back issues*	$3.95	$5.00	$3	$5	$5	_____
UniNews Newsletter subscription	30.00	60.00	8	11	30	_____
1993 *Open Systems Products Directory*	45.00	95.00	7	15	55	_____
1993 UniForum Proceedings	75.00	150.00	7	15	55	_____
Network Substrata	5.00	10.00	2	3	6	_____
Network Applications	5.00	10.00	2	3	6	_____
The UniForum Guide To Graphical User Interfaces	4.95	9.95	2	3	6	_____
Electronic Mail De-Mystified	5.00	10.00	3	4	9	_____
Internationalization Explored (#)	5.00	10.00	3	4	9	_____

(#) Publication size: ☐ 8 1/2 x 11 ☐ 5 1/2 x 8 inches

Calif. residents add applicable sales tax (memberships and postage not taxable) _____

*Specify individual back issues _____. **TOTAL AMOUNT ENCLOSED $**_____

UniForum Mail Lists: Contact a UniForum marketing representative at (408) 986-8840 ext. 26 for information on membership and conference attendee lists.

Send order and payments to:
UniForum
2901 Tasman Drive, Suite 201
Santa Clara, CA 95054
Tel: (408) 986-8840 (800) 255-5620
Fax: (408) 986-1645

UniForum
The International Association of Open Systems Professionals

Prices subject to change without notice. Contact UniForum for discounts and shipping and handling charges on bulk orders.

03.24.93 9361117 (inq)

Report on Fiji Computer Society Conference

March 25 – 27

P. McCrea

I was invited by the Fiji Computer Society (FCS) to present the keynote talk at FIJICOM 93, their Annual Conference, the theme of which was *Open Systems and the Future*. The invitation was made to the AUUG President, so in a very real sense I was speaking on behalf of AUUG. I conveyed official greetings from AUUG to the FCS. It is interesting to note that the FCS got my name from the articles I occasionally write for *Open Systems Review*.

UNIX is not all that pervasive in FIJI, so the brief I was given, was to provide a crash course in both UNIX and Open Systems to the attendees. The keynote address focussed on industry trends, standards, and open system technologies. My second talk was specifically on UNIX, where I gave a spiel on where UNIX has come from, where it is now, and where it appears to be heading, heavily peppered by a beware-of-NT-as-it's-not-an-open-system philosophy!

Apparently the Colonel himself was to have opened the Conference, but he pulled out at the last minute, and was ably represented by the Hon Etuate Tavai, Minister of State, Special Duties – PM's office. The Minister in his opening address said that the Government was currently putting together an IT strategy. As he did in fact stay for my Keynote talk, I hope something of an open systems flavour will be present in their IT strategy.

The three main hardware players in Fiji, who were all present at the conference – both physically and with sponsorship – are DEC, Fujitsu (ICL really), and IBM (via their dealer, Datec). All appear to be pushing UNIX quite vigorously. Conspicuously absent were our colleagues from Microsoft...

I was not the only Australian speaker at the Conference. Rolf Jester from DEC, who is also on the AUUG Management committee, gave a fine presentation, as did Roger Fraumann of UNIX International, who arrived in fine style from the US, where he had been attending Uniforum. (OK, Roger is actually American, but after living here for some time, we can now call him a pseudo–Australian!). Roger had been at the COSE announcement at Uniforum, so he was able to give the audience first hand information.

There was considerable interest on the part of the FCS committee in strengthening the association between AUUG and the FCS. AUUG may pick up a few members from Fiji. At the recent AUUG Management Committee meeting we agreed to offer AUUG membership rates to any member of the FCS who wish to attend AUUG 93 in Sydney in September.

The conference started on a Thursday evening, and ended Saturday evening. The Saturday and Sunday happened to be the weekend of the Hong Kong rugby sevens. What a place to watch it on TV! The Fijians take their rugby _very_ seriously, and when Fiji played Australia in the semi final, it was a good time to put a sock in one's mouth! (One's mouth was a little numb from kava anyway...) The over-powering gloom that overcame the Travelodge, where the conference was held, when Western Samoa beat Fiji in the finals was intense to say the least.

I enjoyed the conference immensely. There is something about Fiji which is relaxing – assisted by the hypnotic rhythm of the near-continuous torrential rain. I believe we have built up a good relationship with the FCS, which we should strive to maintain. For those readers with enquiring minds, the FCS paid for travel and accommodation, and AUUG was not out of pocket a single cent...

Phil McCrea

Four New Chapters for AUUG Inc.

AUUG Inc. is pleased to announce the formation of **four** new chapters. South Australia, Queensland and the Northern Territory have now formed chapters. WAUG (the Western Australian UNIX Group), previously an organisation affiliated with AUUG, has become the Western Australian chapter of AUUG. These chapters were all approved at the April Management Committee meeting. The new chapters join the **two** existing chapters (ACT and Victoria) to give a chapter in each mainland state, except NSW. The formation of a chapter in NSW in imminent. (If you wish to assist in this, contact the AUUG Business Manager.)

All chapters are running a programme of local technical meetings and other events. Details of local chapter events are announced in AUUGN and by mailout to chapter members. If you are interested in participating in a local chapter event or would like more information about the chapters, please contact the chapter representatives shown below.

Chapter meetings are an excellent place to keep track of new technology developments and meet with peers and experts in a wide range of areas. AUUG encourages its members to share their experiences and knowledge through our National Winter Conference, local Summer Conference Series and through regular local chapter meetings. If you have experience or knowledge to share please consider a presentation at an AUUG event.

Chapter Contacts:

	Contact	Phone	Fax.	Email
ACT	John Barlow	(06) 249 2930	(06) 249 0747	john.barlow@anu.edu.au
NT	Mike Clarke	(089) 46 6671	(089) 27 0612	mike.clarke@ntu.edu.au
QLD	Greg Birnie	(07) 340 2111	(07) 340 2100	greg@lna.oz.au
SA	Michael Wagner	(08) 212 2800	(08) 231 0321	
VIC	Neil Murray	(03) 764 1100	(03) 764 1179	neil@wcc.oz.au
WA	Glenn Huxtable	(09) 380 2878	(09) 380 1089	glenn@cs.uwa.edu.au

Peter Wishart
AUUG Inc. - Secretary

SESSPOOLE

AUUG Victorian Chapter

SESSPOOLE is the offical Victorian chapter of AUUG Inc. It was the first Chapter of the AUUG to be formed, and its members have been involved in the staging of the Victorian AUUG Summer technical meetings every year since 1990. SESSPOOLE currently meets approximately every six weeks to hold alternate social and technical meetings. It is open to all members of AUUG Inc., and visitors who are interested in promoting further knowledge and understanding of UNIX and Open Systems within Victoria.

The purpose of the social meetings is to discuss UNIX and open systems, drinking wines and ales (or fruit juices if alcohol is not their thing), and generally relaxing and socialising over dinner. Whilst the technical meetings provide one or two "stand-up" talks relating to technical or commercial issues, or works in progress of open systems.

The program committee invites interested parties wishing to present their work, to submit informal proposals, ideas, or suggestions on any topics relating to Open Systems. We are interested in talks from both the commercial and research communities.

Social meetings are held in the Bistro of the *Oakleigh Hotel, 1555 Dandenong Road, Oakleigh*, starting at about 6:30pm. Venues for the techincal meetings are varied and are announced prior to the event. The dates for the next few meetings are:

Wed, 21 July '93	Technical
Thu, 2 September '93	Social
Tue, 12 October '93	Technical
Wed, 24 November '93	Technical
Thu, 16 December '93	Social
Tue, 24 January '94	Social
Wed, 1 March '94	Technical
Thu, 12 April '94	Social

Hope we'll see you there!

To find out more about SESSPOOLE and its activities, contact the committee or look for announcements in the newsgroup **aus.auug**, or on the mailing list **sesspoole@clcs.com.au**.

SESSPOOLE Committee			
President:	Stephen Prince Chancery Lane Computer Services Phone: (03) 608 0911 Email: sp@clcs.com.au	**Secretary:**	Neil Murray Webster Computer Corporation Phone: (03) 764 1100 Email: neil@wcc.oz.au
Treasurer:	John Carey Labtam Australia Phone: (03) 587 1444 Email: john@labtam.oz.au	**Programme Chair:**	Michael Paddon Iconix Phone: (03) 571 4244 Email: mwp@iconix.oz.au

AUUG Inc – Victorian Chapter
A Review of the Last Few Months

Michael Paddon

May 31, 1993

The Melbourne Summer Conference

The worst part about organising a summer conference is that nagging voice in the back of your mind that says "nobody's going to show up", later replaced by "you'll never get enough people to break even". Such worries were swept from my mind when 61 people from all over Victoria (and even a few from interstate) collected their badges from me that Friday morning, the 26^{th} of February.

This year, the Melbourne Summer Conference was held at the Clunies Ross Conference Center. This marked a watershed in our local conference series, in that it was the first time we felt both the need and the confidence to utilise a professional venue. In retrospect, the decision was correct; the lecture theatre and associated facilities were superb, and the on-site catering was excellent as well as convenient.

The day kicked off with Greg Bond telling us all about stockbroking and open systems. He provided some amusing anecdotes about working for a stockbroking firm that is based on the first floor (they can't jump when the market takes a dive) and, more seriously, he spoke about why a business would choose Unix over PC's, and how they would implement that decision. Another issue Greg tackled was the availability and evolution of different windowing systems in such an environment and how he dealt with the problems generated by "look and feel".

Greg Rose then took the floor and gave a lively report on the 1993 Usenix conference. The major topics he touched on were the Usenix board meeting (which preceeded the conference) and a selection of some of the more interesting papers. What generated the most interest, however, was Greg's description of the legal battle between AT&T and BSDI and the ensuing "mentally contaminated" badges that were ubiquitous at Usenix. Upon concluding his report, Greg mentioned that he had a limited number of the badges to give away...

After the dust settled, we broke for morning tea and resumed with Ken McDonell's detailed excursion into the Unix kernel, the aim being to trace real time events. Ken not only succeeded in explaining the black magic he had performed on his kernel, but he also managed to impart a real feel for how one should go about instrumenting and measuring real time software in general, and how to avoid the associated pitfalls.

Paul-Michael Agapow presented what was definitely the most controversial paper of the day. Drawing on his background in biochemistry, he laid down a strong case for computer viruses acting analogously to terrestrial life forms. This lead forward to his conclusion that viral evolution may not only be possible, but inevitable. The truly disturbing aspect of Paul-Michael's paper was how well the logic hung together; nevertheless, this presentation generated a range of responses that had to be seen to be believed.

The lunch break was followed by Ian Hoyle telling us all about the the new MIME multimedia mail standard. He contrasted MIME with the existing standards, RFC822 and X.400, and explained just what the new standard promised and delivered. Ian went on to explain how one can integrate MIME into an existing complex mail architecture, in this case the one operated by BHP Research.

Douglas Ray continued the networking thread with a presentation on how he goes about gathering AARNet usage statistics at the University of Melbourne. He focussed on exactly which usage figures are useful for network management and how to extract such information from currently existing equipment. Douglas uses a software suite called NNStat for this task, and he explained how this package works, and where one should place monitoring stations in a complex topology.

Following afternoon tea, Richard West was to have spoken about object oriented programming on the NeXT. At the last minute, unfortunately, Richard was prevented from attending. Now, there's nothing quite like trying to find a replacement speaker a day or so before a conference and I found this task to be as impossible as I feared. Luckily, I had a technical report handy that quickly got re-hammered to fit the spare slot – I spoke about the algorithms needed to displaying real world images under X11, concentrating on the problems of colour quantization.

Our last paper of the day was given by Peter Chubb, who extolled the virtues of a truly open system, Linux. Peter described what Linux was (a full featured, POSIX compliant system), how to get it and where it was going. Of especial interest was the list of features that Linux alone has, including some really neat pseudo file systems.

In short, the AUUG Melbourne Summer Conference of 1993 was a strong addition to a string of successful past events. For the fiscally minded, the conference cost $2593.85, with $3140.50 raised in registrations and $300.00 in sponsorship, yielding a profit of $846.65. Looking beyond the bottom line, however, the most valuable contribution was the enthusiasm of the local open systems community in supporting AUUG's summer conference series.

I'd like to take this chance to sincerely thank all of the speakers. I've said it before: it takes hard work and guts to present a paper in front of your peers, and each and every one of the above named people performed a sterling job. Thanks also to Jon Eaves and John Carey; without these guys it all would have fallen apart early on. Steven Lynch and Chris Karadiglis stuffed envelopes beyond and above the call of duty; I owe you one. Finally, my gratitude goes to Silicon Graphics who were kind enough to provide sponsorship for the printing of proceedings.

The SESSPOOLE Annual General Meeting

Immediately following the close of the Summer Conference, the outgoing SESSPOOLE committee convened their 1993 Annual General Meeting. The topics of discussion included:

- the new chapter funding arrangements
- the need for regular technical meetings and their format
- the future directions of SESSPOOLE

The general consensus was to use the monies provided by AUUG to hold a total of four technical meetings a year. These would complement the existing programme of informal social gatherings. The possibility of holding at least one of these meetings outside of Melbourne was also raised.

Following general discussion, elections were held for the new committee. Stephen Prince and John Carey were nominated unopposed to the positions of President and Treasurer, respectively. After some discussion as to the various duties involved, Neil Murray was nominated to the position of Secretary. Michael Paddon was appointed by the new committee as Programme Chair for the technical meetings.

The Inaugural Technical Meeting

April rolled around far quicker than was convenient, but the first technical meeting (on Thursday the 29^{th}) fell into place thanks to the enthusiasm of our speakers and the generous offer from ACCI (Australian Computing and Communications Institute) to provide a venue.

Lim O. Sim told us about a portable C task package that he had written as a tool to assist with producing simulations. Unlike some vendor offerings, it supports truly light weight threads with an elegant means of controlling and communicating between the different contexts.

James Gardiner introduced us to the world of digital effects by describing his experiences with the Abekas A66 real time digital video recorder. This device is of particular interest because of the way it sits on a network and acts just like any other TCP/IP host. This makes the task of pulling down a video frame to your workstation for processing as simple as using FTP.

Both papers were received extremely well, and in each case I had to cut question time short. Following the second talk, we adjourned to a restaurant in nearby Lygon Street, where many meters of pizza were consumed.

As I write this article, the final arrangements for the second technical meeting have not quite been set in concrete. SESSPOOLE members (if you are a Victorian AUUG member, this means you) will be mailed with the final details before too long, and I'll also post them to aus.auug. If you are interested in coming along, and neither of these forums will reach you, please contact me and I'll arrange to put you on our mailing list.

I am also on the look out for people willing to speak for either fifteen minutes or a half and hour on an issue of technical relevance in the open systems arena. There's a lot of people doing interesting things out there; if you're one of them, why not come along to one of our technical meetings and share your experiences?

Michael Paddon
mwp@iconix.oz.au
Phone: (03) 571 4244
Fax: (03) 571 5346

WAUG and Perth News

Greetings from the West to the West and the rest.

I am pleased to report that WAUG is, at last, a formal chapter of AUUG. AUUG's committee have accepted the petition and the necessary arrangements have been made.

On 19 May WAUG held two AGMs (twice the fun, eh?:-( ). The first was a meeting of the old WAUG, which voted to formally dissolve the old organisation. After that, the first formal meeting of the new WAUG elected the chapter committee for 1993/94. Glenn Huxtable is the Chapter Chair, Major is Secretary, and Patrick Ko is Treasurer. The ordinary committee members are Mark Baker, Adrian Booth, Luigi Cantoni, Bernd Felsche, Don Griffiths, and myself.

After the AGMs Adrian gave an excellent ten-minute talk on "Mining for Gold in the Unix Kernel". He has kindly provided some notes, including a useful sample program, for this issue of AUUGN.

While I'm talking of Adrian I would like to congratulate him on his organisation of the successful 1993 Perth Summer Technical Conference. He did particularly well in attracting sponsorship from Sun, Silicon Graphics and Dymocks, and in producing a snazzy set of proceedings. I hope this sets a standard for future conferences.

A good crowd attended the eight professionally-presented talks. Of the talks, I found five riveting, one interesting, one soporific and one incomprehensible. That's a 75% "worth bothering" rating — not bad.

I actually enjoyed giving my tutorial on Perl for Systems Administrators. With the help of a Macintosh (boo hiss:-) and `xfig`, I made up camel nametags for all the attendees. Thanks to O'Reilly and Associates' book cover, the camel is Perl's mascot.

I was presented with a gift voucher, kindly donated by Dymocks, which I promptly spent. They didn't yet have the new O'Reilly, "Unix Power Tools" (which I have since ordered direct from O'Reilly by email — more publishers should provide this service), so I got Terry Pratchett's latest Discworld novel, a book on knots, and a book about the history of the English language.

I'm hoping to present the tutorial again at AUUG'93. With any luck there'll be camel badges all round.

Boss willing, I intend to be at the inaugural SAGE-AU conference (Melbourne, July 7-9), so hopefully I'll see some of you there.

Janet Jackson <janet@cs.uwa.edu.au>
From WAUG, the WA Chapter of AUUG

FYI: WAUG's postal address is PO Box 877, WEST PERTH WA 6005.
Email addresses: waug@uniwa.uwa.edu.au, waug-meetings@uniwa.uwa.edu.au, waug-newsletter@uniwa.uwa.edu.au.

Overview of AUUG Summer Technical Conference 1993 — Perth

The 1993 Perth Summer Technical Conference was the fourth to be held in Perth. It was as successful as the previous ones, featuring two high-calibre interstate speakers (Greg Rose and Chris Schoettle), a low-cost tutorial program, and an informal cocktail event.

Seven talks were presented. Several of the papers have been supplied to AUUGN for publication, so I will keep the description of each talk brief:

Greg Rose (Australian Computing and Communications Institute) gave an entertaining report on the USENIX 1993 Winter Conference.

Chris Schoettle (UNIX System Laboratories, Australia and New Zealand) gave two consecutive talks, the first on SVR4 Enhanced Security, the second on Online Transaction Processing under UNIX.

Paul Templeman (Sequel Technology) described UNIX Network Backup and Archival Strategies.

Harald Reiss (Reiss Dynamics Enterprise) explained Distributed Object Management.

Mike Schulze (Curtin University of Technology) described a suite of network monitoring tools developed as a research project at Curtin. Mike's excellent talk won him the Best Local Speaker prize.

Dr Chris McDonald (University of Western Australia) gave the only talk to include source code — a description of a programming environment that simulates networking protocols developed by students under X Windows.

Greg Rose concluded the conference with another entertaining talk, by re-presenting his Invited Talk at the 1993 USENIX 1993 Winter Conference, "A History of UNIX".

Unfortunately, by then, the conference was running well over time, which meant that only half or so of the attendees stayed for the cocktail event. Also unfortunately, this meant that we had to consume a quantity of food and drink provided for double our number. Fortunately some willing and experienced help was at hand (no names will be mentioned :-).

The cocktail event was generously sponsored by Sun Microsystems. (The highlight of the conference to many was when, during Greg's last talk, he described how Sun and AT&T formed an alliance to unify UNIX — it was precisely at this point that a Sun Microsystems sign fell from the wall with a resounding crash).

A set of conference proceedings was produced, courtesy of the sponsorship of Silicon Graphics. The proceedings were received very well by all attendees.

Dymocks booksellers had an extensive book stall present, and were offering a 10% discount to all attendees. Dymocks sponsored both the Best Local Speaker prize won by Mike Schulze, and a present for Janet Jackson, the only local to present a tutorial. As a result of the conference, Dymocks Hay Street Mall plan to dramatically expand their UNIX-related book stocks — I will follow this with interest.

As mentioned, Janet Jackson gave a tutorial. Those who know Janet will be astounded that it was about *perl*. The tutorial attracted around a dozen attendees, and was an excellent one — especially considering that the price for AUUG members was only $60.

The other tutorial was "System V Release 4 Technical Overview and Selected Internals Topics" by Chris Schoettle. This tutorial spanned a whole day, with the Technical Overview held in the morning and the Internals Topics in the afternoon, and attracted 30 attendees — quite a respectable number.

By lunchtime however, a few people had opted not to stay for the Internals Topics — the morning session had already been more technical than they had expected.

All in all, the conference was a success, and everyone I spoke to said that they would attend again next year. Hope to see you there!

Adrian Booth, Adrian Booth Computing Consultants <abcc@dialix.oz.au>, (09) 354 4936
From WAUG, the WA Chapter of AUUG

Mining for Gold in the UNIX kernel

(I recently gave a talk of this title to the inaugural meeting of the W.A. chapter of AUUG. At the request of several people present I have written this summary of the talk. Note that this article is slanted towards BSD-derived systems).

The UNIX kernel is simply a (quite large) executable program. Like any other program, it consists of instructions ("text") and data ("data" and "bss"). This article summarises some methods of extracting this data from the kernel.

Much of the interesting data within the kernel is kept in fixed-sized tables (arrays of `structs`). Examples of such tables include the process table, the open file table, and the inode table.

UNIX provides several utilities to extract data from the kernel. Perhaps the best known is `ps`. `ps` grovels through the process table and prints out the information that it finds there in a (hopefully) readable format.

Another useful utility is `pstat`. `pstat` allows the contents of other system tables to be dumped in what is often a slightly less readable format. `pstat -T` can be used simply to determine the size and occupancy of each system table (and of swap space).

To understand how programs like `ps` work, we first need to take a step back and examine how executable files are generated.

When a program is compiled, a table describing the *symbols* or references within that program is built, and is written into the executable file. This table usually contains several pieces of data about each symbol - its size, type (text, initialised data, etc.), and location within the memory address space of a process executing the file.

Since the kernel is itself merely an executable file, it too has a symbol table within it.

Programs like `ps` use this symbol table to find out where particular pieces of information (such as the process table) are within the kernel's address space.

They then read the kernel's address space at the specified location to extract the required information. (More on this shortly).

This by the way explains why you should never run `strip` on the kernel file - the purpose of `strip` is simply to remove the symbol table from an executable file. There is in general no real need to run `strip` on any binary unless you are running out of disk space and are *very* desperate for more.

If you need to get information from the kernel that programs like `ps` weren't written to provide, the next step is to learn how to use `adb`. `adb` is a general-purpose assembler debugger that can be used to both examine and to patch executables.

To use `adb` to examine the values of kernel variables, you would invoke it with the command line:

```
adb -k /vmunix /dev/mem
```

Within `adb`, you can examine the value of variables with the **?** command. Commands like **?** expect modifiers after them so `adb` knows how much data to read and in what format to present it. Modifiers include **D** (read a 32-bit number and print it in decimal), **X** (read a 32-bit number and print it in hexadecimal), and **s** (read a NUL-terminated string and display it).

So to find out the size of the process table (typically stored in a variable named `nprocs`), we would say:

```
nprocs?D
```

The **?** command reads the value from the executable file (`/vmunix`). To read the value from kernel memory instead, use the / command. This command accepts the same modifiers as ?, so we could say:

```
nprocs/D
```

to extract the value of `nprocs` from kernel memory.

Warning - here be dragons

You can also use `adb` to patch executables - either the executable file itself, or the memory image of the running image of the executable.

To patch the kernel in this fashion, we would invoke `adb` with a **-w** flag, causing the kernel file and the memory image to be opened in read/write mode:

```
adb -kw /vmunix /dev/mem
```

As an example, to change a 32-bit value in the executable file, the **W** modifier is used to the **?** and / commands:

```
var?W10     # change var's initial value in the executable

var/W10     # change var's current value in memory
```

These patches should only be applied if you know what you are doing *and* have a good set of backups!

A few prerequisite/miscellaneous pieces of information before we continue:

• Our `adb` command lines referred to `/dev/mem` - an image of the physical memory of the machine. `/dev/kmem` is similar, except that it is an image of kernel virtual memory. The `-k` flag tells `adb` to perform kernel memory mapping (we need this since we are running `adb` on the kernel).

• Some versions of UNIX only allow the superuser to access these special memory files. Others define a unique group (such as kmem) and give members of that group read access to these files, so programs such as `ps` can be set-GID kmem instead of set-UID `root`.

• We can find out the names of symbols within the symbol table using the nm command, which simply dumps the symbol table.

• Within the symbol table, symbols are usually preceded by an underscore (_) - a program that refers to a variable `i` would have a corresponding entry for `_i` in its symbol table.

It is surprisingly simple to write programs that can delve into the kernel and even change its operation.

For each piece of kernel data in which we have an interest, we:

• Use a library call to find out its address within kernel memory, based on its name;

• Seek to the returned address within `/dev/kmem`;

• Read the appropriate number of bytes from this position.

This of course assumes that we know the name of the symbol to start with.

The first step - finding the symbol's address - is performed using the `nlist()` library call:

```
#include <nlist.h>
nlist(filename, nl)
char *filename; /* the name of the executable file */
struct nlist *nl;
```

For each symbol we are interested in, we create a `struct nlist` which has its `n_name` member pointing to a string containing the name of the symbol.

So to find the address of the `nprocs` symbol within `/vmunix`, we would use the following code fragment:

```
struct nlist nl[2];

nl[0].n_name = "_nprocs";
nl[1].n_name = NULL;

nlist("/vmunix", nl);
```

Assuming that the call succeeds, we can now examine the `n_type` and `n_value` members of `nl[0]` to find out about the symbol.

If we have already opened `/dev/kmem` and have a file descriptor `kd` that references it, we next:

```
lseek(kd, nl[0].n_value, SEEK_SET);
```

Finally (assuming that `nprocs` is an integer), we can read its value into our program:

```
int numprocs;

read(kd, &numprocs, sizeof(int));
```

A sample program fleshing out these steps has been appended to this article. It takes two arguments - the name of the kernel file, and the name of the kernel symbol we want to examine. For simplicity, it assumes that the symbol refers to an integer.

We can easily expand on the techniques in this sample program to extract information from more complex kernel data structures - the system's header files are an excellent source of information about these structures, and can be "reverse engineered".

These sorts of programs tend to get very messy - it simplifies things remarkably to write a function that accepts a kernel offset, a memory address, and a byte count, and copies the specified number of bytes from the kernel offset to the memory address.

Two traps for young players that I must mention:

• Pointers within kernel structures refer to *kernel* addresses, *not* to addresses within your program. Dereference these pointers by `lseek()`ing to their value within `/dev/kmem`.

• The user "u dot" area can be swapped out with a process - try not to do much with it when it is swapped out!

Under preparation - part 2, covering areas such as `libkvm`, `/proc` filesystems, and more!

Adrian Booth, Adrian Booth Computing Consultants <abcc@dialix.oz.au>, (09) 354 4936
From WAUG, the WA Chapter of AUUG

```
#include <stdio.h>
#include <nlist.h>
#include <unistd.h>
#include <sys/file.h>
#include <sys/types.h>
#include <sys/uio.h>

main(argc, argv)
int  argc;
char **argv;
{
   struct nlist  nl[2];
   int    kd;
   char   *kernel;
   char   *kvar;
   int    value;

   if (argc != 3) {
      fprintf(stderr, "Usage: %s kernel variable\n", argv[0]);
      exit(1);
   }

   kernel = argv[1];
   kvar = argv[2];

   nl[0].n_name = kvar;
   nl[1].n_name = NULL;

   if (nlist(kernel, nl) < 0) {
      perror("nlist");
      exit(1);
   }

   if (nl[0].n_type == N_UNDF) {
      fprintf(stderr, "%s is not defined in %s\n", kvar, kernel);
      exit(1);
   }

   printf("%s offset = %08x\n", kvar, nl[0].n_value);

   if ( (kd = open("/dev/kmem", O_RDONLY)) < 0 ) {
      perror("open");
      exit(1);
   }

   if (lseek(kd, (off_t) nl[0].n_value, SEEK_SET) != nl[0].n_value ) {
      perror("lseek");
      exit(1);
   }

   if (read(kd, (char *) &value, sizeof(value)) < 0) {
      perror("read");
      exit(1);
   }

   printf("%s = %d\n", kvar, value);

   exit(0);
}
```

SAGE-AU Inaugural Conference and Annual General Meeting

Dates: Wednesday 7 to Friday 9, July 1993

Venue: The University of Melbourne

Preliminary Announcement and Call for Papers

The System Administrators Guild of Australia (SAGE-AU) will be hosting a conference in conjunction with its inaugural annual general meeting. The theme of the conference will be

"Administering Networked Computers"

With the coming of age of computer networks, more and more organisations have internal networks of machines sharing information. Administration of these machines involves solving far more complex problems than for standalone computers. System administrators are under increasing pressure to allow more and more interconnection of machines without any loss of reliability or security.

SAGE-AU'93 solicits papers on all aspects of computer administration, particularly on the problems and solutions of administering networked computers.

Conference Details

SAGE-AU'93 will be a 3 day conference running from Wednesday the 7th to Friday the 9th of July, 1993.

The first day will be dedicated to tutorials on tools and techniques to aid system administration.

The inaugural AGM will be held at the end of the second day, July 8.

All other times will be allocated to presentations. A conference dinner will be held on the second night.

The conference will feature a small trade show on the second and third days focusing on system administration tools.

Tutorials

Tutorial sessions will be either half day or full day duration. People wishing to present tutorials should submit an abstract and a preference for a half day or full day slot to the address below. Tutorials should be run in a lecture format.

Suggested topics include:

- NIS/NIS+ administration
- Automount/AMD
- PERL as a administration tool
- Network backup strategies and techniques
- Kerberos
- Password Security
- Sendmail configuration
- AppleTalk/IP Interoperation
- IP Networking with PC's
- Electronic Mail gateways

Note that due to time constraints, proposals to present tutorials must be submitted by 7th May, 1993.

Presenters will be paid $200 for a half day tutorial and $400 for a full day tutorial and will receive free conference registration. SAGE-AU will reimburse presenters for reasonable costs of handout materials.

Attendance at tutorials will cost $100 for a half day and $200 for a full day.

Papers

Slots are available for 15 minute, 30 minute and 60 minute presentations. 5 minutes should be reserved for questions from the audience.

15 minute slots are less formal and are present to allow people to talk briefly about some topic of interest, administration problem they are having or have solved without having to prepare a formal paper.

People presenting papers in the 30 and 60 minute slots will receive a 50% discount on registration fees.

If you wish to present a paper, send an abstract to the address below. Please indicate whether you wish to use a 15, 30 or 60 minute slot.

Abstracts should be 100 - 200 words in length.

Papers should have a technical orientation and not contain advertising.

Deadlines

Applications to present tutorials must reach the organisers by May 7, 1993.

Applications to present papers must reach the organisers by May 15, 1993.

Registration

Registration forms are available from the address below. People registering before May 31 receive a 25% discount on the registration fee.

Fees

The registration fee for SAGE-AU members is $100.
Non-members may register for $150.

The registration fee does NOT include the conference dinner. Registrants must indicate whether they wish to attend the conference dinner on the registration form and pay an additional cost. In keeping with the low-cost, informal nature of the conference, the conference dinner may be held in a local restaurant, depending on numbers.

The registration fee does not include attendance at tutorials.

SAGE-AU foundation membership is currently being offered for $30 ($20 annual subscription plus a $10 joining fee).

Annual membership fees have yet to be set by the committee.

Addresses

Send all enquiries regarding the conference to

Peter Gray
Professional Officer
Dept of Computer Science
University of Wollongong
N.S.W. 2500 Australia

EMail: pdg@cs.uow.EDU.AU

Phone: +61 42 213 770
Fax: +61 42 213 262

Requests for general information about SAGE-AU and membership applications should be addressed to

Frank Crawford
Australian Supercomputer Technology
Woods Centre
PMB 1
Menai NSW 2234

or emailed to sage-info@mel.dit.csiro.au.

System Administrators Guild Conference - 1993

Tutorial Topics

SAGE-AU'93 will be offering the following tutorials on Wednesday, July 7.

Setting and maintaining Internet firewalls - full day

Making the SOLARIS 2 environment work for the system adminsitrator 1/2 day, 9:00AM - 12:00 noon

Perl for Systems Administrators 1/2 day, 1:00PM - 4:00PM

Cost is $100 per person for either of the 1/2 day tutorials, and $200 per person for the full day tutorial.

Full details of the tutorials are given below.

People wishing to obtain a tutorial registration form or additional information on SAGE-AU please contact:

Peter Gray
Dept of Computer Science
University of Wollongong
Wollongong NSW 2566

Email: pdg@cs.uow.edu.au
Phone: 042 213770
Fax: 042 213262

Title: Setting and maintaining Internet firewalls
Duration: Full day
Presenter: Brent Chapman

Target Audience

The target audience for this tutorial includes network and system managers who are contemplating construction of an Internet firewall security system, or are maintaining an existing firewall system.

Attendees should already understand basic Internet networking principles; for example, they should understand packet encapsulation, IP addressing, and the differences between UDP and TCP.

While oriented toward sites contemplating connection to (or already connected to) the Internet, much of the information in the tutorial is applicable to intra-organization networking where different parts of the network have different security needs.

Tutorial Description

Many valuable resources are accessible to sites on the Internet, such as fast, convenient electronic mail to vendors, clients, and colleagues; discussion groups on a wide range of topics; and vast archives of freely usable software.

Unfortunately, there is also a dark underside to the Internet: individuals and groups that, for a variety of motives including fun and profit, will take advantage of lax security at sites on the Internet.

In the "good old days", most sites had few computers, and even fewer of these computers had access to the Internet; security efforts concentrated on securing each machine individually. Today, however, networked computers are the norm rather than the exception. Many modern high-tech companies have more networked computers than employees. Securing all these systems one-by-one is usually impractical, for a variety of reasons. Fortunately, with modern networking technology, it's possible to take a more collective approach to security, and concentrate on securing entire networks of machines, rather than individual machines.

An Internet firewall system might be defined as a system that lets a site take advantage of some of the services offered on the Internet (such as electronic mail and anonymous FTP), while at the same time limiting the site's exposure to attack from the Internet.

This tutorial teaches you how to build a firewall between your site and the Internet. The class starts with a look at the problems that a firewall attempts to address, then proceeds to an analysis of different types of firewall systems. The tutorial examines packet filtering in particular as a means of firewall construction, and concludes by working through the design and construction of a firewall system based on packet filtering.

This tutorial is very practical in nature, and includes many examples and anecdotes. It provides information and insights valuable across a wide range of installation sizes (from a single-system operation through a multi-thousand-node networked site), operation types (including academic, research, corporate, and government), and platforms (such as personal computers, UNIX workstations, shared computing resources, local-area and wide-area networks, internetworks, and so forth), with a concentration on networked UNIX workstations.

Biography

Brent Chapman is a consultant in the San Francisco Bay Area, specializing in the networking of UNIX systems. He has built many Internet firewall systems for a wide variety of clients, using a range of techniques and technologies. He is the manager of the "Firewalls" Internet mailing list, and has taught previous USENIX tutorials on "Preparing for Disaster". Before founding Great Circle Associates, he was an operations manager for a financial services company, a world-renowned corporate research lab, a software engineering company, and a hardware engineering company. He holds a Bachelor of Science degree in Electrical Engineering and Computer Science from the University of California, Berkeley.

Title: Perl for Systems Administrators
Duration: 1/2 day
Presenter: Janet Jackson

Target Audience

Attendees should be systems administrators comfortable with Unix tools and concepts and having at least a reading knowledge of shell and of C or another high-level language. No prior knowledge of Perl is required.

Tutorial Description

Perl is a powerful, freely available scripting language that fills a niche between shell- and C-level programming. Its capacity for text-processing and system interaction make it an excellent systems administration tool.

This tutorial introduces Perl with an emphasis on systems administration, by examining practical programs. After the tutorial, attendees will be able to start solving some of their own systems administration problems using Perl.

Biographical sketch

Janet is a systems administrator in the Department of Computer Science at the University of Western Australia.

She has been using Perl for systems administration since about 1990 --- before the Camel book came out, anyway, as she remembers learning Perl from the manpage.

At AUUG Summer '92 Perth she gave a paper on automating user administration with Perl. The paper was subsequently published in AUUGN.

She has used Perl to develop a time management suite and lots of other useful programs, many of which will be shown in the tutorial.

Title: Making the SOLARIS 2 environment work for the system adminsitrator
Duration: 1/2 day

Target Audience

Actual or potential system administraors of SUN computers either using or planning to use the SOLARIS 2 operating system.

Tutorial Description

This tutorial will describe the system administration environment of SOLARIS 2. The various administartion tools shipped with SOLARIS 2 will be covered including: admin tool, host manager, database manager, printer manager, software manager, jumpstart and NIS+.

Third party administration tools will also be discussed.

Biographical Details

This tutorial will be presented by three SUN employees.

Fraser Gardiner: has been with Sun for over 4 years and in the UNIX environment for 8 years. He is Systems Engineer Manager for the Victorian branch of SUN Microsystems, Australia.

Rich Baxter: has been in the computer industry for 14 years and with Sun for 5 years in the areas of professional services and systems engineer specialising in graphics technology.

Dennis Letizia: has been in the computer industry for 12 years and currently has the role within Sun of networking and telecommunications specialist. Dennis was also involved in the worldwide Solaris technical training.

Know Your UNIX System Administrator – A Field Guide

by Stephan Zielinski
<szielins@us.oracle.com>

There are four major species of UNIX sysad:

1) *The Technical Thug*. Usually a systems programmer who has been forced into system administration; writes scripts in a polyglot of the Bourne shell, *sed*, *C*, *awk*, *perl*, and APL.

2) *The Administrative Fascist*. Usually a retentive drone (or rarely, a harridan ex-secretary) who has been forced into system administration.

3) *The Maniac*. Usually an aging cracker who discovered that neither the Mossad nor Cuba are willing to pay a living wage for computer espionage. Fell into system administration; occasionally approaches major competitors with bizarre schemes.

4) *The Idiot*. Usually a cretin, morphodite, or old COBOL programmer selected to be the system administrator by a committee of cretins, morphodites, and old COBOL programmers.

How To Identify Your System Administrator

Situation: Low disk space

Technical Thug: Writes a suite of scripts to monitor disk usage, maintain a database of historic disk usage, predict future disk usage via least squares regression analysis, identify users who are more than a standard deviation over the mean, and send mail to the offending parties. Places script in *cron*. Disk usage does not change, since diskhogs, by nature, file all mail away in triplicate without reading it.

Administrative Fascist: Puts disk usage policy in *motd*. Uses disk quotas. Allows no exceptions, thus crippling development work. Locks accounts that go over quota.

Maniac:
```
# cd /home
# rm -rf 'du -s * | sort -rn | head -1 |
awk '{print $2}'';
```

Idiot:
```
# cd /home
# cat 'du -s * | sort -rn | head -1 | awk
'{ printf "%s/*\n", $2}'' | compress
```

Situation: Excessive CPU usage

Technical Thug: Writes a suite of scripts to monitor processes, maintain a database of CPU usage, identify processes more than a standard deviation over the norm, and *renice* offending processes. Places script in *cron*. Ends up renicing the production database into oblivion, bringing operations to a grinding halt, much to the delight of the *xtrek* freaks.

Administrative Fascist: Puts CPU usage policy in *motd*. Uses CPU quotas. Locks accounts that go over quota. Allows no exceptions, thus crippling development work, much to the delight of the *xtrek* freaks.

Maniac:
```
# kill -9 'ps -augxww | sort -rn +8 -9 |
head -1 | awk '{print $2}''
```

Idiot:
```
# compress -f 'ps -augxww | sort -rn +8 -
9 | head -1 | awk '{print $2}''
```

Situation: New account creation

Technical Thug: Writes perl script that creates home directory, copies in incomprehensible default environment, and places entries in */etc/passwd*, */etc/shadow*, and */etc/group* (by hand, NOT with *passmgmt*). Slaps on *setuid* bit; tells a nearby secretary to handle new accounts. Usually, said secretary is still dithering over the difference between 'enter' and 'return'. No new accounts are ever created.

Administrative Fascist: Puts new account policy in *motd*. Since people without accounts cannot read the *motd*, nobody ever fulfills the bureaucratic requirements; so, no new accounts are ever created.

Maniac: "If you're too stupid to break in and create your own account, I don't want you on the system. We've got too many idiots on this box anyway."

Idiot:
```
# cd /home; mkdir "Bob's home directory"
# echo "Bob Simon:gandalf:0:0::/dev/
tty:compress -f" > /etc/passwd
```

Situation: Root disk fails

Technical Thug: Repairs drive. Usually is able to repair filesystem from boot monitor. Failing that,

† This is a re-print from ;login, the USENIX Association Newsletter, Volume 18 Number 2

front-panel toggles microkernel in and starts script on neighboring machine to load binary boot code into broken machine, reformat and reinstall OS. Lets it run over the weekend while he goes mountain climbing.

Administrative Fascist: Begins investigation to determine who broke the drive. Refuses to fix system until culprit is identified and charged for the equipment.

Maniac, Large System: Rips drive from system, uses sledgehammer to smash same to flinders. Calls manufacturer, threatens lawsuit. Abuses field engineer s while they put in a new drive and reinstall the OS.

Maniac, Small System: Rips drive from system, uses ball-peen hammer to smash same to flinders. Calls Requisitions, threatens pets. Abuses bystanders while putting in new drive and reinstalling OS.

Idiot: Doesn't notice anything wrong.

Situation: Poor network response

Technical Thug: Writes scripts to monitor network, then rewires entire machine room, improving response time by 2%. Shrugs shoulders, says, "I've done all I can do," and goes mountain climbing.

Administrative Fascist: Puts network usage policy in *motd*. Calls up Berkeley and AT&T, badgers whoever answers for network quotas. Tries to get *xtrek* freaks fired.

Maniac: Every two hours, pulls Ethernet cable from wall and waits for connections to time out.

Idiot: `# compress -f /dev/en0`

Situation: User questions

Technical Thug: Hacks the code of *emacs*' doctor-mode to answer new users questions. Doesn't bother to tell people how to start the new "guru-mode", or for that matter, *emacs*.

Administrative Fascist: Puts user support policy in *motd*. Maintains queue of questions. Answers them when he gets a chance, often within two weeks of receipt of the proper form.

Maniac: Screams at users until they go away. Sometimes barters knowledge for powerful drink and/or sycophantic adulation.

Idiot: Answers all questions to best of his knowledge until the user realizes few UNIX systems support punched cards or JCL. --

Situation: *Stupid* user questions

Technical Thug: Answers question in hex, EBCDIC, and/or French until user gives up and goes away.

Administrative Fascist: Locks user's account until user can present documentation demonstrating their qualification to use the machine.

Maniac:

```
 # cat >> ~luser/.cshrc alias vi 'rm
\!*;unalias vi;grep -v BoZo ~/.cshrc > ~/
.z; mv -f ~/.z ~/.cshrc' ^D
```

Idiot: Answers all questions to best of his knowledge. Recruits user to become part of system administration team.

Situation: Religious war, BSD vs. System V

Technical Thug: BSD. Crippled on System V boxes.

Administrative Fascist: System V. Horrified by the people who use BSD. Places frequent calls to DEA whenever requests for BSD features are made.

Maniac: Prefers BSD, but doesn't care as long as *his* processes run quickly.

Idiot: # cd c:

Situation: OS upgrade

Technical Thug: Reads source code of new release, takes only the modules he likes.

Administrative Fascist: Instigates lawsuit against the vendor for having shipped a product with bugs in it in the first place.

Maniac:
```
# uptime
1:33pm up 19 days, 22:49, 167 users, load
average: 6.49, 6.45, 6.31
# wall
```
Well, it's upgrade time. Should take a few hours. And good luck on that 5:00 deadline, guys! We're all pulling for you!
^D

Idiot: `# dd if=/dev/rmt8 of=/vmunix`

Situation: Balky mail

Technical Thug: Rewrites *sendmail.cf* from scratch. Rewrites *sendmail* in SNOBOL. Hacks kernel to implement file locking. Hacks kernel to implement "better" semaphores. Rewrites *sendmail* in assembly. Hacks kernel to . . .

Administrative Fascist: Puts mail use policy in

motd. Locks accounts that go over mail use quota. Keeps quota low enough that people go back to interoffice mail, thus solving problem.

Maniac:

```
# kill -9 'ps -augxww | grep sendmail |
awk '{print $2}''
# rm -f /usr/spool/mail/*
# wall
```

Mail is down. Please use interoffice mail until we have it back up.

```
 ^D
# write max
```

I've got my boots and backpack. Ready to leave for Mount Tam?

```
^D
```

Idiot:

```
# echo "HELP!" | mail tech_support.AT.v-
endor.com%kremvax%bitnet!BIFF!!!
```

Situation: Users want phone list application

Technical Thug: Writes RDBMS in perl and Smalltalk. Users give up and go back to post-it notes.

Administrative Fascist: Oracle. Users give up and go back to post-it notes.

Maniac: Tells the users to use flat files and *grep*, the way God meant man to keep track of phone numbers. Users give up and go back to post-it notes.

Idiot:

```
# dd ibs=80 if=/dev/rdisk001s7 | grep
"Fred"
```

Hobbies, Technical

Technical Thug: Writes entries for Obsfuscated C contest. Optimizes INTERCAL scripts. Maintains ENIAC emulator. Virtual reality.

Administrative Fascist: Bugs offices. Audits card-key logs. Modifies old TVs to listen in on cellular phone conversations. Listens to police band.

Maniac: Volunteers at Survival Research Labs. Bugs offices. Edits card-key logs. Modifies old TVs to listen in on cellular phone conversations. Jams police band.

Idiot: Ties shoes. Maintains COBOL decimal to roman numeral converter. Rereads flowcharts from his salad days at Rand.

Analogies to Live Without: You don't need a manual to drive a car.

by Elizabeth Zwicky
<zwicky@erg.sri.com>

When people start complaining about how complex computers are, they often assert that manuals should be unnecessary – after all, you don't need a manual to drive a car. You can rent a car of a make you've never driven, get in it, and drive it away.

There are two basic problems with this assertion. First, there's an initial learning curve difference. When I learned to drive a car, I had been riding in cars for 15 years; I had a solid base of relevant knowledge gained from riding bicycles; and I was trained by a 3 month process involving books, classroom instruction, films, simulators, and many, many hours of practice, all of it with someone present to coach me. During this process, I used various different cars, in order to help me learn to transfer these skills from car to car. When I learned to use a computer, I had only seen it done once or twice before, and I was plopped down in front of one with a book and another novice. Furthermore, this casual approach actually worked; using only a book and my wits, I learned to make the computer do what I wanted. I don't believe that this would have been possible with the car.

I am now more or less able to make *any* computer work, from an IBM PC to a IBM 370, using only a book. On the other hand, the only cars I can drive are automatic transmission passenger cars; without another training process, I can't drive a motorcycle or a truck.

Second, it's simply not true that you don't need a manual to drive a new car. I have been in a car with three other reasonably intelligent computer professionals, at a gas station, unable to open the gas tank for the several minutes that it took us to figure out how to open the glove compartment to get at the manual. (As it turned out, the manual wasn't in the glove compartment, but the release for the gas cap was, much to our surprise.) I once had to refer to the manual to figure out how to get the key out of the ignition, in a car with the same manufacturer and model year as the car I normally drove. Sure, you can generally start the car, and with a little diligence you can usually work out how to turn on the lights and run the windshield wipers (particularly if it's light out and not raining), and you can almost always get the seatbelts to work, but what good does that do you if you can't put gas in it?

At that, cars are much better in this respect than a lot of the other appliances we work with. Nobody ever asserts that computers should be more like VCRs, because almost everybody has a VCR at home blinking "12:00." Half the coffee-makers I've seen are effectively impossible to use without a manual, and three of the seven buttons on the little CD player I use at work do incomprehensible modal things. Compared to this, UNIX starts to look real good.

I'm not saying that current computer interfaces are paragons of human engineering; I swear at my computer plenty. But let's make fair comparisons. Cars have been around for about 100 years now, and computers for less than half that. The tasks that cars perform are extremely well understood and cover only a very small range – they move around in two dimensions, and they keep people inside reasonably safe while doing so. Computers, on the other hand, perform a wide range of dissimilar tasks. Finally, cars are incredibly familiar objects, which most Americans have experience with starting only days after they are born. With all these advantages, we still put vastly more effort into teaching people to drive than into teaching them to use computers. Why is it the computer interface that people say is hard to use?

For one thing, you only have to learn to drive once. Cars are so standardized within types that you can buy a new car without a large extra increment of learning. You have to look at a manual, but not for long. This is partly because they've been around longer, and partly because they are specialized to a small range of tasks. You do have to go through a longer learning period if you need a vehicle that has new capabilities; that may be a short time, if you're adapting to a van or an off-road vehicle, or a long time, if you're adapting to an 18-wheeler. Second, so much of people's time is spent around cars that they are effectively experts. People experience different cars all the time, and even when they aren't actually driving, they're picking up information about the cars they're in. Third, driver's education is a cultural norm. Everybody knows what it takes to learn to drive, whether they've done it or only seen it done on TV. Fourth, it seems only fair that it takes a while to learn to do something that is so dangerous. Several tons of steel, garnished with protective devices, inspires respect.

Computer training doesn't transfer as easily from one machine to the next, most people aren't computer experts, computer training isn't something that everybody does, and computer accidents rarely kill people. Thus, people actually hold computers to a higher usability standard than cars. Computers actually do pretty well, considering how flexible and how new they are. It would be nice if they did better, but cars are not a standard for improvement.

System Administration Tools Your Vendor Never Told You About: The Laminator

by Elizabeth Zwicky
<zwicky@erg.sri.com>

A laminator is about $300 worth of completely noncomputerized electrical equipment. It takes flat things that involve pieces of plastic, and heats them up while squishing them. Different laminators allow different configurations, but the general principle is that you put a piece of paper between two pieces of plastic, run it through the machine, and come out with a solid block of plastic with the paper embedded in it. (This is the gizmo vendors use at trade shows to turn your business card into a luggage tag.)

The point of this is that it produces signs, tags, and stickers that say whatever you want but withstand age and mishandling well. In combination with a laser printer, it produces slick-looking signs and notices. (In combination with a color printer, it produces absolutely amazing signs.) We laminate all of our signs (including the ones that we move around, like "This equipment under repair. Do not attempt to use or reassemble.") and any instructions that we want to put next to equipment. As well as full sheets, you can laminate tags for cables, and you can make labels. The printing wears off laser-printed labels quite fast, but laminated labels last forever, and since you're printing plain paper they don't jam in the printer.

The Ten Commandments for C Programmers †

(Annotated Edition)

by Henry Spencer
<henry@zoo.toronto.edu>

Thou shalt run *lint* frequently and study its pronouncements with care, for verily its perception and judgement oft exceed thine.

This is still wise counsel, although many modern compilers search out many of the same sins, and there are often problems with *lint* being aged and infirm, or unavailable in strange lands. There are other tools, such as Saber C, useful to similar ends.

"Frequently" means thou shouldst draw thy daily guidance from it, rather than hoping thy code will achieve *lint's* blessing by a sudden act of repentance at the last minute. De-linting a program which has never been *linted* before is often a cleaning of the stables such as thou wouldst not wish on thy worst enemies. Some observe, also, that careful heed to the words of *lint* can be quite helpful in debugging.

"Study" doth not mean mindless zeal to eradicate every byte of *lint* output – if for no other reason, because thou just canst not shut it up about some things – but that thou should know the cause of its unhappiness and understand what worrisome sign it tries to speak of.

Thou shalt not follow the NULL pointer, for chaos and madness await thee at its end.

Clearly the holy scriptures were mistranscribed here, as the words should have been "null pointer," to minimize confusion between the concept of null pointers and the macro NULL (of which more anon). Otherwise, the meaning is plain. A null pointer points to regions filled with dragons, demons, core dumps, and numberless other foul creatures, all of which delight in frolicing in thy program if thou disturb their sleep. A null pointer doth *not* point to a 0 of any type, despite some blasphemous old code which impiously assumes this.

Thou shalt cast all function arguments to the expected type if they are not of that type already, even when thou art convinced that this is unnecessary, lest they take cruel vengeance upon thee when thou least expect it.

A programmer should understand the type structure of his language, lest great misfortune befall him.

Contrary to the heresies espoused by some of the dwellers on the Western Shore, 'int' and 'long' are not the same type. The moment of their equivalence in size and representation is short, and the agony that awaits believers in their interchangeability shall last forever and ever once 64-bit machines become common.

Also, contrary to the beliefs common among the more backward inhabitants of the Polluted Eastern Marshes, NULL does not have a pointer type, and must be cast to the correct type whenever it is used as a function argument.

(The words of the prophet Ansi, which permit NULL to be defined as having the type 'void *', are oft taken out of context and misunderstood. The prophet was granting a special dispensation for use in cases of great hardship in wild lands. Verily, a righteous program must make its own way through the Thicket Of Types without lazily relying on this rarely-available dispensation to solve all its problems. In any event, the great deity Dmr who created C hath wisely endowed it with many types of pointers, not just one, and thus it would still be necessary to convert the prophet's NULL to the desired type.)

It may be thought that the radical new blessing of "prototypes" might eliminate the need for caution about argument types. Not so, brethren. Firstly, when confronted with the twisted strangeness of variable numbers of arguments, the problem returns... and he who has not kept his faith strong by repeated practice shall surely fall to this subtle trap. Secondly, the wise men have observed that reliance on prototypes doth open many doors to strange errors, and some indeed had hoped that prototypes would be decreed for purposes of error checking but would not cause implicit conversions. Lastly, reliance on prototypes causeth great difficulty in the Real World today, when many cling to the old ways and the old compilers out of desire or necessity, and no man knoweth what machine his code may be asked to run on tomorrow.

---

† This is a re-print from ;login, the USENIX Association Newsletter, Volume 18 Number 2

If thy header files fail to declare the return types of thy library functions, thou shalt declare them thyself with the most meticulous care, lest grievous harm befall thy program.

The prophet Ansi, in her wisdom, hath added that thou shouldst also scourge thy Suppliers, and demand on pain of excommunication that they produce header files that declare their library functions. For truly, only they know the precise form of the incantation appropriate to invoking their magic in the optimal way.

The prophet hath also commented that it is unwise, and leads one into the pits of damnation and subtle bugs, to attempt to declare such functions thyself when thy header files do the job right.

Thou shalt check the array bounds of all strings (indeed, all arrays), for surely where thou typest "foo" someone someday shall type "supercalifragilisticexpialidocious".

As demonstrated by the deeds of the Great Worm, a consequence of this commandment is that robust production software should never make use of *gets()*, for it is truly a tool of the Devil. Thy interfaces should always inform thy servants of the bounds of thy arrays, and servants who spurn such advice or quietly fail to follow it should be dispatched forthwith to the Land Of Rm, where they can do no further harm to thee.

If a function be advertised to return an error code in the event of difficulties, thou shalt check for that code, yea, even though the checks triple the size of thy code and produce aches in thy typing fingers, for if thou thinkest "it cannot happen to me," the gods shall surely punish thee for thy arrogance.

All true believers doth wish for a better error-handling mechanism, for explicit checks of return codes are tiresome in the extreme and the temptation to omit them is great. But until the far-off day of deliverance cometh, one must walk the long and winding road with patience and care, for thy Vendor, thy Machine, and thy Software delight in surprises and think nothing of producing subtly meaningless results on the day before thy Thesis Oral or thy Big Pitch To The Client.

Occasionally, as with the *ferror()* feature of *stdio*, it is possible to defer error checking until the end when a cumulative result can be tested, and this often produceth code which is shorter and clearer. Also, even the most zealous believer should exercise some judgement when dealing with functions whose failure is totally uninteresting... but beware, for the cast to void is a two-edged sword that sheddeth thine own blood without remorse.

Thou shalt study thy libraries and strive not to re-invent them without cause, that thy code may be short and readable and thy days pleasant and productive.

Numberless are the unwashed heathen who scorn their libraries on various silly and spurious grounds, such as blind worship of the Little Tin God (also known as "Efficiency"). While it is true that some features of the C libraries were ill-advised, by and large it is better and cheaper to use the works of others than to persist in re-inventing the square wheel. But thou should take the greatest of care to understand what thy libraries promise, and what they do not, lest thou rely on facilities that may vanish from under thy feet in future.

Thou shalt make thy program's purpose and structure clear to thy fellow man by using the One True Brace Style, even if thou likest it not, for thy creativity is better used in solving problems than in creating beautiful new impediments to understanding.

These words, alas, have caused some uncertainty among the novices and the converts, who knoweth not the ancient wisdoms. The One True Brace Style referred to is that demonstrated in the writings of the First Prophets, Kernighan and Ritchie. Often and again it is criticized by the ignorant as hard to use, when in truth it is merely somewhat difficult to learn, and thereafter is wonderfully clear and obvious, if perhaps a bit sensitive to mistakes.

While thou might think that thine own ideas of brace style lead to clearer programs, thy successors will not thank thee for it, but rather shall revile thy works and curse thy name, and word of this might get to thy next employer. Many customs in this life persist because they ease friction and promote productivity as a result of universal agreement, and whether they are precisely the optimal choices is much less important. So it is with brace style.

As a lamentable side issue, there has been some unrest from the fanatics of the Pronoun Gestapo over the use of the word "man" in this Commandment, for they believe that great efforts and loud shouting devoted to the ritual purification of the language will somehow redound to the benefit of the downtrodden (whose real and grievous woes tendeth to get lost amidst all that thunder and fury). When preaching the gospel to the narrow of mind and short of temper, the word

"creature" may be substituted as a suitable pseudo-Biblical term free of the taint of Political Incorrectness.

Thy external identifiers shall be unique in the first six characters, though this harsh discipline be irksome and the years of its necessity stretch before thee seemingly without end, lest thou tear thy hair out and go mad on that fateful day when thou desirest to make thy program run on an old system.

Though some hasty zealots cry "not so; the Millenium is come, and this saying is obsolete and no longer need be supported," verily there be many, many ancient systems in the world, and it is the decree of the dreaded god Murphy that thy next employment just might be on one. While thou sleepest, he plotteth against thee. Awake and take care.

It is, note carefully, not necessary that thy identifiers be limited to a length of six characters. The only requirement that the holy words place upon thee is uniqueness within the first six. This often is not so hard as the belittlers claimeth.

Thou shalt foreswear, renounce, and abjure the vile heresy which claimeth that "All the world's a VAX," and have no commerce with the benighted heathens who cling to this barbarous belief, that the days of thy program may be long even though the days of thy current machine be short.

This particular heresy bids fair to be replaced by "All the world's a Sun," or "All the world's a 386" (this latter being a particularly revolting invention of Satan), but the words apply to all such without limitation. Beware, in particular, of the subtle and terrible "All the world's a 32-bit machine," which is almost true today but shall cease to be so before thy resume grows too much longer.

One Day at School †

by Jeff West
< *jwest@els.cray.com*>

I'm taking the CS 480 course, Operating System Principles, at the U. this semester. Last night the professor got into explaining the UNIX *fork()* call and one of the students just couldn't get it through his head that fork actually makes a copy of the process you are running (thus yielding two processes).

The code that he was using to demonstrate was as follows:

```
if(fork() != 0)
{
   printf("I'm the parent.\n");
}
else
{
  printf("I'm the child.\n");
}
```

The student/professor banter went something like:

S: So the kernel will return the process ID and we'll print "I'm the parent."

P: Correct.

S: But if it returns a 0 we print, "I'm the child."

P: Correct.

S: So which one will it return?

P: Both.

[looooooong pause]

S: But the kernel can only return one value.

P: Correct.

S: Well which one does it return?

P: Both.

[looooooooooooooonger pause]

Abbott and Costello would have been proud.

† This is a re-print from ;login, the USENIX Association Newsletter, Volume 18 Number 2

Australian Users' Views on Open Systems

In this report:

Datapro Summary

The second of a series of informal end-user discussions organized by Datapro International took place in Sydney in mid-November 1992 . Five members of the Australian UNIX Users Group (AUUG) and the Australian User Alliance for Open Systems (AUAOS) sparked frank conversation about the implementation of open systems in a production environment.

Mediated by Dr. Philip McCrea, president of the AUUG, the closed-door session revealed the realities of an "open" choice. While "open systems" now exist for hardware choices, it is applications software and specifically the database management system and development tools that actually drive the decision.

Defining Open Systems

Participants debated the validity of the term "open systems" as they searched for a common denominator in its definition.

Peter Thomas, operations support manager of the New South Wales Land Titles Office, views open systems in the context of the current information technology revolution. "It is the ability to take business information from a mainframe system and have it dynamically linked to a spreadsheet on an executive's desk, or being able to exchange documents across the world," said Thomas. "Changing the emphasis to organizational data is what open systems represents to me."

Another participant defined open systems as "the creation of a standard and the compliance with that standard over time."

Some participants regarded open systems to be no more than marketing hype. They felt that many of the systems sold today as "open" were the proprietary systems of yesterday.

The vendor community plays an important role in the move towards open systems. For instance, TCP/IP is a standard that does not belong to any vendor, yet forms the basis of most commercially available open systems communications. TCP/IP became a standard only because the vendor community was willing to accept and began supporting it in their products. Open systems started to gather momentum in the last few years as vendor commitment to provide interoperability increased.

The Real Benefits

One of the main benefits of moving from proprietary to open systems is cost. With open systems, users can avoid being locked in to a hardware vendor. Based on standards, open systems allows users to mix and match systems without worrying about the problem of incompatibility. Users can therefore select the most cost effective products which best suit their needs.

In addition, users do not have to depend on a single vendor for all its needs. With proprietary systems, users continually need to go back to the same system vendor to expand their systems, add software and peripherals and for

— By Patricia Dewar Striplin
Freelance Writter

maintenance. A participant said, "Ten years ago, running an IBM shop with an IMS database on MVS meant that we could only communicate using SNA, nothing else."

Open systems also bring about the possibility of running off-the-shelf software packages. With availability of an increasing number of packaged software for open systems, users now have a wide selection to choose from, even for some specialized applications. To many users, this option can help them beat the high cost of running a full-time in-house software development team.

The cost benefits of not being locked in to one hardware vendor are particularly significant in the area of maintenance. Very often, maintenance accounts for about two-thirds of the total software price. Unlike proprietary software, packaged software can be maintained by the third-party as well as the developer. Maintenance costs can, therefore, be substantially reduced as users will not be at the mercy of the developer.

Apart from savings in costs, open systems can lead to higher worker productivity in an organization. Open systems permit greater connectivity among different machines and allow users to share and exchange data more effectively. Dr. Glynn Peady, a participant from the Australian Nuclear Science & Technology Organization (ANSTO), said open systems implementation has allowed his organization to fully reap the benefits of its existing network infrastructure.

Open systems is also "a good fall back strategy." Geoff Shoults, MIS manager at ANI Manufacturing Ltd, related the open system advantage in his corporation where he is constantly faced with disparate systems. "If you get it wrong, if the strategy or direction changes, or if there is an unexpected occurrence, you are not stuck," he said. "Previously you would have been. Open systems doesn't solve the integration problem, but it is a hell of a benefit to have."

Shoults added that the open systems movement gives users a chance to voice out their real needs to the vendor community. In a proprietary system environment, system implementation is often based on what the vendor has to offer rather than the user's needs. "Open systems have opened the doors to users .... giving them more input into higher level decision making, and more strategic decisions which require greater consensus."

Supporters of Open Systems

In Australia, IT consultancy firms seem to be the primary supporters of open systems. Because of the nature of their business, consultants are in a good position to recommend a move to or away from proprietary platforms.

An indication of Australia's increasing support for open systems is a demographic change in AUUG membership over the last two to three years. According to Dr. Phil McCrea, a recent AUUG study found there are now an equal number of technical and non-technical members in AUUG. "More recent members are interested in open systems per se, rather than UNIX as a piece of technology," Dr. McCrea said. "As a result, AUUG is becoming a vocal user organization sharing ideas amongst members."

Within the Australian Commonwealth government, the case for open systems is being advanced through the Information Exchange Steering Committee (IESC). With members drawn from major government departments, IESC works to actively promote open systems standards such as GOSIP.

At the state level, several governments act as advisors for information technology architecture. The Queensland state government seems to be the most proactive in promoting UNIX and open systems. It has specified computer systems procurement of open systems for several years now.

The participants conceded that probably the greatest push towards open systems comes from the newspapers and trade journals. As users constantly gain new information about the advantages of open systems, they make demands for openness, portability and connectivity. In addition, top executives who read about the trend of implementing open systems may apply pressure for their own IT departments to look into open systems. "It is easier to sell open systems to executive management regardless of what it is," said one participant.

Impediments to Open Systems

A cost-benefit analysis between proprietary and open systems may clearly show the cost effectiveness to migrate to open systems. However, there was consensus among the participants that the initial implementation requires a significant amount of effort. Data modeling and operating system utilities for a production center need more planning than in the very mature IBM environment.

The participants agreed that the best long-term direction was open systems but they supported a more gradual conversion process over four to five years, replacing systems as the need arises.

Dr. McCrea pointed out that the standards making processes are tedious and slow. As a result, users still find a limited range of open system products currently available in the market. "If it were a benevolent dictatorship, we would have more products now," he remarked.

As global standards take a long time to evolve, the impatient user community has prompted vendors to enhance products merely to satisfy their demands. Because there is still a lack of standards, vendors resort to creating proprietary extensions to existing standards. In a way, this may prove to be beneficial as proprietary extensions such as those for SQL and X-Windows will eventually be pushed toward the standards committees.

On the other hand, some users view proprietary extensions as a tactic by larger vendors to slow down the move to open system standards. "Some vendors are still trying to lock the customer in by recommending proprietary extensions," commented a participant.

While open systems are now available for hardware choices, the lock-in and major cost today is the database management system (DBMS). Standards are not tight enough at the database level to provide portability. Each DBMS has proprietary extensions which may enhance its functionality at the cost of openness. Participants indicated that applications software touted as "open" may be portable across hardware platforms, but not across DBMS platforms. This severely limits hardware choices as certain critical applications can run only on one DBMS.

Although open systems bring promises of lower costs, DBMS software is still equally expensive on the open systems platform. ANI's Shoults commented, "At one level we have an open systems benefit but in the area where we're paying the most money, we haven't seen the benefit yet."

Hardware and software selection were easy to gauge in the proprietary environment. However, in the open systems selection process, benchmarks for software performance are almost non-existent. Participants tended to first follow a "software" approach — establishing a variety of weightings based on what the users

Opportunities for Exports

An open systems platform is more than portability and scalability. For two Australian UNIX User Group (AUUG) members, it also brings export revenue.

Softway, a UNIX system software house founded in 1986, is now a US$2 million company. It developed a kernel level UNIX resource management technology called "Share" which has been licensed to Cray, Convex, Pyramid and Fujitsu.

"Since Australia has no computer manufacturers, Softway saw a niche in attracting the best local UNIX talent and presenting their collective expertise overseas," said Dr. Phil McCrea, Softway's managing director.

Softway also undertook the kernel-level development of UXPM, Fujitsu Ltd's mainframe version of System V.4. Fujitsu, Softway's largest customer, acquired 40% ownership in the company in July 1991.

At the New South Wales Land Titles office, its in-house developed software and expertise presented an opportunity for exports. Based on the recommendations of an independent study, the department migrated from a MVS/XA environment using IMS with COBOL, to an open systems platform with a relational database management system (RDBMS). The new platform integrates all applications and communications. As the Land Titles office enjoyed the benefits of cost savings, more corporate control, and better use of information, it saw an opportunity to export its applications software and expertise.

The land records management application integrates cadastre information, (public record, value, and ownership) with map storage and an imaging system replacing microfiche. Future plans call for a graphical database integrating land data coordinates and all other relevant information. A pilot system was launched in November 1992 for training and to develop procedures for UNIX systems management prior to moving into a full-scale production environment.

Since 40% of the department's revenues are derived from remote users such as banks, government departments and solicitors performing title searches, Land Titling will be the first module re-written for the open systems environment.

Recently, the NSW Land Titles Office partnering the Queensland state government and BHP Technology won a tender funded by the World Bank to study the feasibility of an Automated Land Title System for Indonesia. Several former Soviet Bloc countries have also visited the Sydney site as they look to leap-frog technology in the business of land titling. As a result of the department's international export efforts, the Land Titles office's open systems project was cited as a model of IT development by the NSW's Department of State Development.

wanted. They then applied technical criteria requesting performance statistics on the associated applications and DBMS software prior to final hardware selection. These performance specifications surfaced as a major roadblock for open systems hardware acquisition.

Describing this tendering experience, Dr. Peady says that every vendor's response had "no chance of meeting the performance requirements of that tender because they [hardware vendors] had no idea of the load an ordinary DBMS puts on their system."

Clarke Gerber, computer services manager at New South Wales Maritime Services Board, feels that implementing open systems is not an arduously difficult task. "You must have a good corporate data model and know what you want to do," he said.

With the move from proprietary to open systems, the software life cycle has shortened considerably from an average of 20 years to about five years. While software on proprietary systems "is made to last", users expect lower maintenance, re-useability, greater end-user flexibility and shorter software life cycles in an open systems environment. Gerber believes the software design process should be rethought to gain the greatest performance from 'C' and open systems. An advocate of rewriting applications, he added, "It is wrong to just convert from a 3GL to a 4GL database if users do not gain the greatest advantages from the conversion."

Old perceptions of file system corruptibility and the lack of security features on open systems negatively impact acceptance in the IS technical arena. An awareness of new file journaling software and the introduction of volume management should rectify most file system concerns. Open systems are no less secure than other operating systems, according to Dr. McCrea. But, the high connectivity of open systems, particularly in a wide area networking environment, require extra security precautions.

The participants lamented the lack of sophisticated software tools such as utility software, systems management, recovery and backup software on the open systems platform. These tools are inherent to proprietary systems and they are highly sophisticated in VMS and MVS production environments. While vendors have committed to port utility software to their open systems-based hardware, in the meantime users experience the pain of a less efficient operation.

The high cost of systems integration is also an impediment to open systems implementation. As a result, one participant's organization made a strategic decision to support only one portable corporate database, eliminating the costs associated with multiple DBMS upkeep. Furthermore, many users, particularly the small and medium-sized enterprises, lack the expertise to effectively mix and match the most cost effective products for a truly open computing platform.

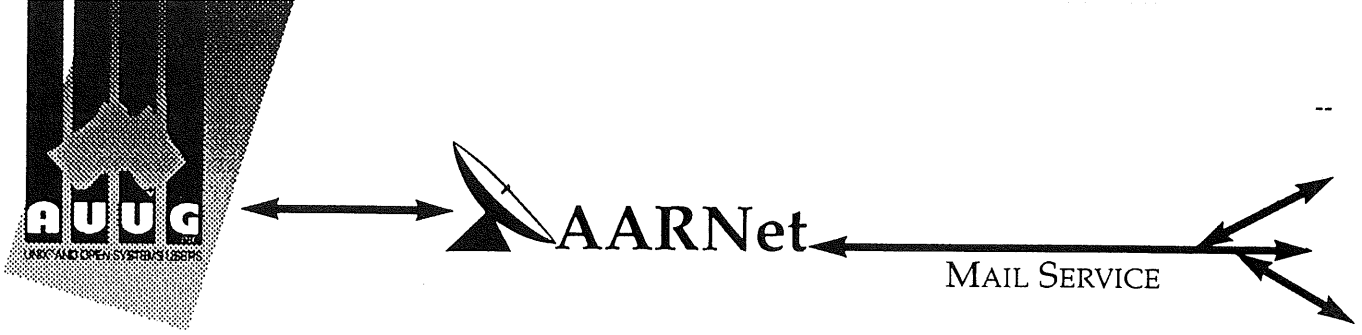
Conclusion

While the user community reached consensus on the long term value of migration to open systems, mixed feelings prevailed about the speed of change. Companies valuing the longer term

effects of open systems expect financial benefits in maintenance reduction and the use of packaged applications. They predict greater connectivity and shortened development cycles will bring value to the business through increased responsiveness and competitiveness. Some participants felt the value lies not in the operating system, but in whether information can be exchanged, interoperability can be achieved and other operating systems can be integrated into the user environment.

Implementation concerns centered around the still immature utilities critical to a successful production operation. They expressed the need for open systems vendors to take a more proactive role in developing traditional system utilities.

The round-table participants concluded that the opportunity to choose an open platform for hardware and applications software exists. But, they have found that DBMS extensions create another proprietary environment. To more fully benefit from openness, the user community feels greater emphasis must be placed on performance measurements and standards at the DBMS level. ■



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).

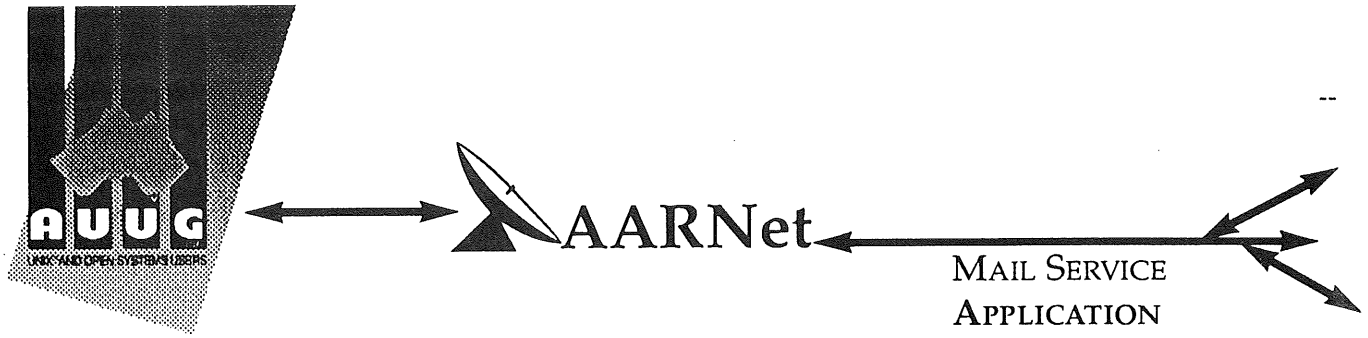
The fee is intended to cover use of AARNet bandwidth for your network traffic. Registration with AARNet, however, provides ONLY the registration of your address in worldwide address tables - your site will be unreachable without this registration. The fee does NOT cover the costs involved in obtaining 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 Liz Fraumann at the AUUG office on (02) 361 5994 (eaf@swift.sw.oz.au) or myself (chris@softway.sw.oz.au).

Regards,
Chris Maltby
AUUG-AARNET Administrator
AUUG Inc.



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.

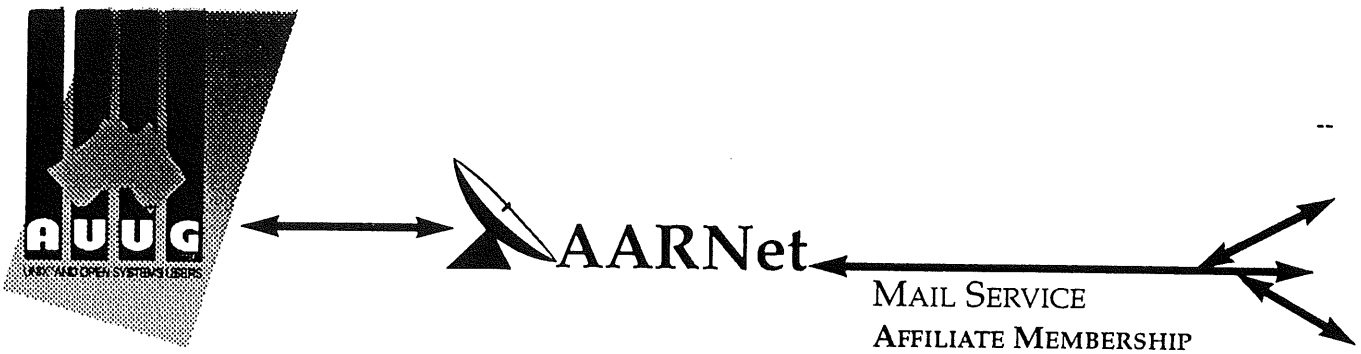
I understand that the AVCC may consult the recorded logs of my organisation's usage of AARNet facilities for 1990, and determine that I am ineligible for registration under the terms of the agreement between AVCC and AUUG Inc. I understand that AUUG Inc will invoice my organisation for this service for 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

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:_____

_____

_____

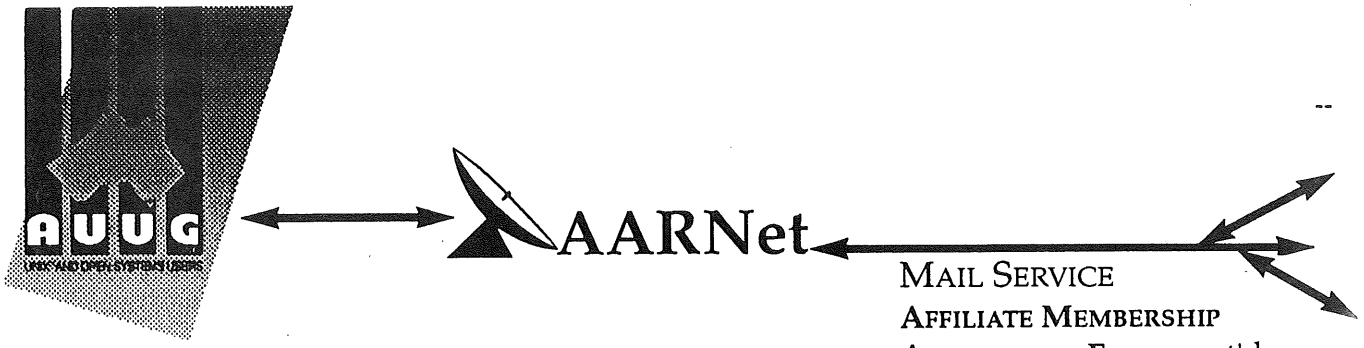
Expected Link Protocol: UUCP SL/IP MHSnet Other:_____

* * * *

Send this page only to:

AUUG Incorporated
PO Box 366
Kensington NSW 2033

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



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.

ACSnet Survey

1.1 Introduction

ACSnet is a computer network linking many UNIX hosts in Australia. It provides connections over various media and is linked to AARNet, Internet, USENET, CSnet and many other overseas networks. Until the formation of AARNet it was the only such network available in Australia, and is still the only network of its type available to commercial sites within Australia. The software used for these connections is usually either SUN III or SUN IV (or MHSnet). For the purposes of this survey other software such as UUCP or SLIP is also relevant.

At the AUUG Annual General Meeting held in Melbourne on September 27th, 1990, the members requested that the AUUG Executive investigate ways of making connection to ACSnet easier, especially for sites currently without connections. This survey is aimed at clearly defining what is available and what is needed.

Replies are invited both from sites requiring connections and sites that are willing to accept connections from new sites. Any other site that has relevant information is also welcome to reply (*e.g.* a site looking at reducing its *distance* from the *backbone*).

Please send replies to:

Mail: Attn: Network Survey
AUUG Inc
P.O. Box 366
Kensington N.S.W. 2033

FAX: (02) 332 4066
E-Mail: auug@atom.lhrl.oz

Technical enquiries to:

Michael Paddon (mwp@iconix.oz.au) (03) 571 4244
or
Frank Crawford (frank@atom.lhrl.oz) (02) 717 9404

Thank you

======

1.2 Contact Details

Name: _____
Address: _____
_____
_____
Phone: _____
Fax: _____
E-Mail: _____

1.3 Site Details

Host Name: _____
Hardware Type: _____
Operating System Version: _____
Location: _____
_____
_____

Host Name: _____

New Connections

If you require a network connection please complete the following section.

Please circle your choice (circle more than one if appropriate).

A1. Do you currently have networking software? Yes No

A2. If **no**, do you require assistance in selecting a package? Yes No

A3. Are you willing to pay for networking software? Yes No
If **yes**, approximately how much? _____

A4. Do you require assistance in setting up your network software? Yes No

A5. Type of software:
SUNIII MHSnet UUCP
TCP/IP SLIP
Other (Please specify): _____

A6. Type of connection:
Direct Modem/Dialin Modem/Dialout
X.25/Dialin X.25/Dialout
Other (Please specify): _____

A7. If **modem**, connection type:
V21 (300 baud) V23 (1200/75) V22 (1200)
V22bis (2400) V32 (9600) Trailblazer
Other (Please specify): _____

A8. Estimated traffic volume (in KB/day): (not counting netnews)
< 1 1-10 10-100
> 100: estimated volume: _____

A9. Do you require a news feed?
Yes No
Limited (Please specify): _____

A10. Any time restrictions on connection?
Please specify: _____

A11. If the connection requires STD charges (or equivalent) is this acceptable? Yes No

A12. Are you willing to pay for a connection (other than Telecom charges)? Yes No
If **yes**, approximately how much (please also specify units, *e.g. $X/MB* or flat fee)? _____

A13. Once connected, are you willing to provide additional connections? Yes No

A14. Additional Comments:

Host Name: _____

Existing Sites

If you are willing to accept a new network connection please complete the following section.

Please circle your choice (circle more than one if appropriate).

B1. Type of software:
SUNIII MHSnet UUCP
TCP/IP SLIP
Other (Please specify): _____

B2. Type of connection:
Direct Modem/Dialin Modem/Dialout
X.25/Dialin X.25/Dialout
Other (Please specify): _____

B3. If **modem**, connection type:
V21 (300 baud) V23 (1200/75) V22 (1200)
V22bis (2400) V32 (9600) Trailblazer
Other (Please specify): _____

B4. Maximum traffic volume (in KB/day): (not counting netnews)
< 1 1-10 10-100
> 100: acceptable volume: _____

B5. Will you supply a news feed?
Yes No
Limited (Please specify): _____

B6. Any time restrictions on connection?
Please specify: _____

B7. If the connection requires STD charges (or equivalent) is this acceptable?
Yes No

B8. Do you charge for connection?
If **yes**, approximately how much (please also specify units, *e.g. $X/MB* or flat fee)?
Yes No
_____

B9. Any other restrictions (*e.g.* educational connections only).?

B10. Additional Comments:

AUUG Book Reviews

The following book reviews have been supplied for this issue of AUUGN. Please note the Prentice Hall order form at the end of the book reviews.

If you would like more details about book reviews see page 5.

The Internet Message
The Fourth book in the networking trilogy.....

by Marshall T. Rose
Prentice Hall, 1993

Reviewed by
Nick Frisina

It was a pleasure to read this book, which is packed with information, author's bug fixes, and believable speculation in an easy to read style.

Rose commences with a history of mail and the Internet, moves on to basic Naming and Addressing, where he separates the message "envelope" from the transport and the content. It is here that Rose makes the point of castigating those programmers in the "family" who diddle with the "headers" (envelope), or content (body) and believes that they should be forced to type in 1000 times:

I will write robust code.

The sections on Multi-Media (MHN) and Privacy Enhanced Mail (PEM) are enlightening. Rose points out that, even as we move forward, emphasising the need for some sort of "standards" in mail headers, that all seem to have forgotten some sort of standard and comprehensible "error message format" that would prove of some use to all.

MIME (Multi-Purpose Internet Mail Extensions), really comes out a winner in this book. Apart from having an extensible architecture which could incorporate error reports and PEM, MIME allows (p230):

— The message body to have nested contents

— The user agent to select among alternative representations of the content (voice, picture, video etc etc)

— for a content to be a pointer to data stored elsewhere (even on another continent...)

— each content can contain arbitrary data, binary or text.

MHS comes out a little worse on two counts. One, as being a culprit in altering the envelope, and secondly, as it "...does not supply enough information to allow for interoperable exchanges of multi-media contents" (p325).

The chapter on Gateways compares the theory to the real world situation. "...in a perfect world..." the gateways should translate the envelope, the headers and body from one mail domain to another, never comingling the 3 objects. "In practice, mail gatewaying is an absolute mess." (p294)

The appendices offer a detailed slamming of OSI (Open Systems Interconnection) for delivering 2 incompatible schemes in 4 years with no improvement in sight. Rose makes the point that:

"The non-OSI base is growing faster than the OSI market"(p329)

and suggests that the IP (Internet Protocol) is the one spreading faster.

The Glossary is detailed, References are numerous and Index comprehensive.

The dedication "for Wrong is Right" is made clear on page 300.

I believe that the text formatting error on page 192 could have been avoided. Finally I'm left wondering about the Americanism to "root" my mail whereas I'd prefer it was routed.

UNIX System V Print Service Administration

by Sally A. Browning (Ed.)
Prentice Hall, 211pp., (Soft Cover)
ISBN: 0-13-016403-8

Reviewed by
Frank Crawford
Australian Supercomputing Technology
<frank@atom.ansto.gov.au>

This book is one of the System Administrator Collection from UNIX Press (a Prentice Hall Title), and is intended to *demystify* the UNIX System V Release 4 Print Service software for system administrators. It brings together all the relevant information in one place, covering such topics as:

— printer configuration,

— customisation,

— administration,

— details of forms and filters, and

— troubleshooting tips.

The main sources it draws from are the UNIX System Administrator's Guide and Reference Manuals, both for the Standard and for Intel systems. The SVR4 Print Service software is unarguably the most complicated of the standard spooling systems, providing network, PostScript, error recovery, forms and filters support, and certainly needs good documentation. Unfortunately, this isn't it. While the information in this book is correct, there is nothing in addition to what is found in the various manuals it is derived from. In fact, aside from the description of the individual commands, most of the information is the same as found in the *System Administrator's Guide.*

For this book to be of any value it would need to provide information beyond what is found in the *UNIX SVR4 System Administrator's Guide.* For a couple of standard problems I certainly didn't find anything additional in this book. You should note that it does provide marginally more information than that found in the *UNIX SVR4 System Administrator's Guide for Intel Processors*, which concentrates mainly on the *sysadm* software interface, but even then it isn't much more.

As the function of this book is to provide information to system administrators, a better alternative for them is to skip this book and buy the *UNIX SVR4 System Administrator's Guide.* The guide will give you the same information about the Print Service software with the added advantage of containing information on other system software.

Managing Projects with make

by Andrew Oram and Steve Talbott
O'Reilly & Associates, Inc., 1991 (2nd Ed),
136pp., (Soft Cover)
ISBN: 0-937175-90-0

Reviewed by
Frank Crawford
Australian Supercomputing Technology
<frank@atom.ansto.gov.au>

This is a new edition of one of the original Nutshell Handbooks. It describes of one of the major Unix development tools, *make*, covering such topics as:

— writing *makefiles*,

— shell variables,

— internal macros,

— suffix rules,

— maintaining libraries, and

— invoking *make* recursively.

The version of *make* covered is the augmented version shipped with System V, although other versions are referenced.

Although this is a short book (it can be read in a weekend), it is packed full of useful information, tips and tricks, that are invaluable to any Unix programmer. In fact there are tips on how to use *make* for non-programmers, *e.g.* to maintain documentation. No matter how small or large the project (from a couple of files, to one involving a number of people and a number of directories) examples are given which can be built upon.

Aside from instruction on how to write *makefiles* there is a detailed description on *make*'s command line arguments, and a description of the common error messages and how to fix them. Finally, there is a description of all the common variants, where they are found, their differences and tests that can be run if you are not sure of your version of make.

In the final analysis, this is a book that should be in the library of everyone involved with Unix, programming or otherwise.

UNIX installation security and integrity, *

by FERBACHE, D. and SHEARER, G. (1992):
Blackwell Scientific Publications, Oxford,
240pp., $96.00 (paperback)

Reviewed by
Tim McGrath
Transport EDI Services, Fremantle
From WAUG, the WA Chapter of AUUG

The first thing that should be pointed out is that the title of this text is "Unix installation security and integrity", not "Unix — Installation, Security and Integrity". It is a text about the security of Unix-based computer systems — I am not sure why "Installation" was chosen for the title and "Integrity" rates only passing reference.

* This review was originally published in the Australian Computer Journal.

The second point is that this is a technical publication, although attempts are made to address the novice Unix-ophile (if that is the word). The authors do not attempt to answer the questions of "Why?" or "How much?" security. Instead they have taken a hands-on, "How to" approach. As such it presents an erratic view on the world of Unix system security and administration.

I managed to glean some useful information from this book but I am not sure it was worth the effort! The chapter on Programming Security was well covered as was the section on Cryptography (although this is not specifically a Unix issue). Overall, I believe the text adds little to already published works.

The authors obviously have considerable experience and knowledge of administration of Unix systems (and in particular SunOS workstations). Unfortunately, the content of this text is patchy and tends to focus on fine details and loose track of the main theme. I must confess that the section on Process Control left me feeling this was indeed a complex web being woven.

This is definitely one for collectors only. A pity because I feel the authors had a real contribution to make, but lacked editorial guidance.

There is a dark feeling in the back of my mind that the problems with this book could actually be transferred by the casual reader to the subject matter itself. Is it that Unix security is a complex, fragmented and inconsistent mixture of proprietary variations and idiosyncrasies? I, for one, do not believe this to be the case.

The subject can be handled in a more practical and understandable manner and I would commend Garfinkel and Spafford (1991)† as the superior work.

Finally, I should mention that the text's presentation was poor, proof reading was inadequate and several paragraphs were duplicated. Nearly 20% of this book is appendices, mostly taken from the Internet publications. Given the target audience, I thought this unnecessary "padding" — references would have been adequate. Perhaps this would also have made the cost more reasonable!

---

† Garfinkel, Simson and Spafford, Gene (1991): Practical UNIX Security, O'Reilly and Associates, USA

HUMOUR

The Story of a Degenerate †

Unix was a program gone bad.

Born into poverty, its parents, the phone company, couldn't afford more than a roll of teletype paper a year, so Unix never had decent documentation and its source files had to go without any comments whatsoever. Year after year, Papa Bell would humiliate itself asking for rate increases so that it could feed its child. Still, unix had to go to school with only two and three letter command names because the phone company just couldn't afford any better. At school, the other operating systems with real command names, and even command completion, would taunt poor little Unix for not having any job or terminal management facilities or for having to use its file system for interprocess communication and locking.

Then, bitter and emasculated by its poverty, the phone company began to drink. During lost weekends of drunken excess, it would brutally beat poor little Unix about the face and neck. Eventually,Unix ran away from home. Soon it was living on the streets of Berkeley. There, Unix got involved with a bad crowd. Its life became a degrading journey of drugs and debauchery. To keep itself alive, it sold cheap source licenses for itself to universities which used it for medical experiments. Being wantonly hacked by an endless stream of nameless, faceless undergraduates, both men and women, often by more than one at the same time, Unix fell into a hell-hole of depravity.

And so it was that poor little Unix began to go insane. It retreated steadily into a dream world, the only place where it felt safe. It took heroin and dreamed of being a real operating system. It took LSD and dreamed of being a raspberry flavoured three-toed yak. It liked that better. As Unix became increasingly attracted to LSD, it would spend weekends reading Hunter Thompson and taking cocktails of acid and speed while writing crazed poetry in which it found deep meaning but which no one else could understand:

```
$sed <$mf >$mf.new -e '1,/^#
    AUTOMATICALLY/!d'
make shlist || ($echo "Searching for .SH files..."; \
    $echo *.SH | $tr ' ' '\012' | $egrep -v '\*' >.shlist)
if $test -s .deptmp; then
    for file in `cat .shlist`; do
        $echo `$expr X$file : 'X\(.*\).SH'`: $file
        config.sh \; \
                /bin/sh $file >> .deptmp
done
$echo "Updating $mf..."
$echo "# If this runs make out of memory, delete
    /usr/include lines." \>> $mf.new
$sed 's|^\(.*\.o:\) *\(.*/.*\.c\) *$|\1 \2; '"$defrule \2|"
    .deptmp \>>$mf.new else
make hlist || ($echo "Searching for .h files..."; \
    $echo *.h | $tr ' ' '\012' | $egrep -v '\*' >.hlist)
$echo "You don't seem to have a proper C
preprocessor. Using grep instead."

$egrep '^#include ' `cat .clist` `cat .hlist` >.deptmp
$echo "Updating $mf..."
<.clist $sed -n\
-e '/\//{\


-e  's|^\(.*\)/\(.*\)\.c|\2.o: \1/\2.c; '"$defrule \1/\2.c|p"
\
-e  d
\
-e '}'
\
-e 's|^\(.*\)\.c|\1.o: \1.c|p' >> $mf.new
<.hlist $sed -n 's|\(.*/\)\(.*\)|s= \2= \1\2=|p' >.hsed
<.deptmp $sed -n 's|c:#include "\(.*\)".*$|o: \1|p' | \
    $sed 's|^[^;]*/||' | \
    $sed -f .hsed >> $mf.new
<.deptmp $sed -n 's|c:#include <\(.*\)>.*$|o:
    /usr/include/\1|p' \>> $mf.new
<.deptmp $sed -n 's|h:#include "\(.*\)".*$|h: \1|p' | \
$sed -f .hsed >> $mf.new
<.deptmp $sed -n 's|h:#include <\(.*\)>.*$|h:
    /usr/include/\1|p' \>> $mf.new
for file in `$cat .shlist`; do
    $echo `$expr X$file : 'X\(.*\).SH'`: $file config.sh \;
    \/bin/sh $file >> $mf.new
done
fi
```

Eventually, Unix began walking down Telegraph Avenue talking to itself, saying "Panic: freeing free inode," over and over again.Sometimes it would accost perfect strangers and yell "Bus error (core dumped)!" or "UNEXPECTED INCONSISTENCY: RUN FSCK MANUALLY!" at them in a high pitched squeal like a chihuaua with amphetamine psychosis. Upstanding citizens pretended it was invisible. Mothers with children crossed to the other side of the street.

Then one evening Unix watched television, an event which would change its life. There it discovered professional wrestling and knew that it had found its true calling. It began to take huge doses of corticosteroids to build itself up even bigger than the biggest of the programs which had beaten it up as a child. It ate three dozen pancakes and four dozen new features for breakfast each day. As the complications of the steroids grew worse, its internal organs grew to the point where Unix could no longer contain them. First the kernel grew, then the C library, then the number of daemons. Soon one of its window systems was requiring two megabytes of swap space for each open window. Unix began to bulge in strange, unflattering places. But Unix continued to take the drugs and its internal organs continued to grow. They grew out its ears and nostrils. They placed incredible

† This is a re-print from DECUS news Volume 15 Number 1

stresses on Unix's brain until it finally liquefied under pressure. Soon Unix had the mass of Andre the Giant, the body of the Elephant Man, and the mind of a forgotten Jack Nicholson character.

The worst strain was on Unix's mind. Unable to assimilate all the conflicting patchworks of features it had ingested, its personality began to fragment into millions of distinct, incompatible operating systems. People would cautiously say "good morning Unix. And who are we today?" and it would reply "Beastie" (BSD), or "Domain", or "I'm System III, but I'll be System V tomorrow." Psychiatrists laboured for years to weld together the two major poles of Unix's personality, "Beasty Boy", an inner-city youth from Berkeley, and "Belle", a southern transvestite who wanted a to be a woman. With each attempt, the two poles would mutate, like psychotic retroviruses, leaving their union a worthless blob of protoplasm requiring constant life support remain compatible with its parent personalities.

Finally, unbalanced by its own cancerous growth, Unix fell into a vat of toxic radioactive wombat urine, from which it emerged, skin white and hair green. It smelled like somebody's dead grandmother. With a horrible grin on its face, it set out to conquer the world.

This article submitted by a person who wishes to remain nameless.
I will forever protect my sources - Ed note

Open System Publications *

As a service to members, AUUG will source Open System Publications from around the world. This includes various proceeding and other publications from such organisations as

AUUG, UniForum, USENIX, EurOpen, Sinix, *etc.*

For example:

EurOpen Proceedings		USENIX Proceedings	
Dublin	Autumn'83	C++ Conference	Apr'91
Munich	Spring'90	UNIX and Supercomputers Workshop	Sept'88
Trosmo	Spring'90	Graphics Workshop IV	Oct'87

AUUG will provide these publications at cost (including freight), but with no handling charge. Delivery times will depend on method of freight which is at the discretion of AUUG and will be based on both freight times and cost.

To take advantage of this offer send, in writing, to the AUUG Secretariat, a list of the publications, making sure that you specify the organisation, an indication of the priority and the delivery address as well as the billing address (if different).

AUUG Inc.
Open System Publication Order
PO Box 366
Kensington, NSW, 2033
AUSTRALIA
Fax: (02) 332 4066

* Please note the UniForum Publications Order Form on page 13.

The following paper is re-printed from AUUGN Volume 2 Number 6. It was presented by the current programme committee chairman at AUUG80. Piers, would it be accepted for AUUG93?

Share Scheduling Works!

Piers Lauder

Basser Department of Computer Science
Sydney University

ABSTRACT

The Share Sceduling Algorithm on Unix provides a per-user scheduler on top of the standard per-process scheduler. It acts to provide equitable allocation of the machine between classes of users, and between users of the same class, according to their allocation of "shares" of the machine.

Introduction

At Sydney University, we have implemented a version of the Cambridge Share Scheduling Algorithm as proposed by J. Larmouth of Salford University, and modified by Andrew Hume of AGSM. This scheduler has been very successful in allocating the limited resources of our student teaching machine (overcoming the otherwise unrestricted generosity of Unix) and has also been surprisingly successful as an advertisment for the sophistication of our teaching service amongst people who might otherwise consider Unix to be a toy operating system unsuited for the real world.

Implementation

The Share algorithm allocates a share of the machine to each user based on his history of past usage, his current working rate, and the ratio of his allocated shares to those of other users concurrently using the machine.

The algorithm as implemented was extensively modified by Hume (as described by Hume in his paper "A Share Scheduler for Unix"). This was mainly due to Larmouth's algorithm being batch oriented, whereas an interactive environment requires quicker responses.

The low level scheduler in Unix allocates use of the cpu amongst competing processes based on their priority which decays very quickly. Thus every proces competes on an equal basis. It is possible for a user to increase his share of the cpu by running many "asynchronous" processes simultaneously. In order to allocate the cpu between users, regardless of the number of

processes, the low level scheduler was changed to use a per-user priority which is added to each process's short term priority. This per-user priority is manipulated by the Share scheduler to affect a user's long term share of the machine.

To calculate the cost of the user's use of the machine, separate charges are made for cpu time, memory occupancy, terminal I/O, backing store I/O, and system services. This cost is added to the accumulating usage, and the accumulating rate. The rate figure decays quite fast, approximately 10% second, whereas the usage decays quite slowly, approximately 10% day. There are thus two separate figures contributing to the calculations for the real share of the machine to be allocated in the next time period, one based on recent working rate, and the other on long term usage. It is therefore possible for high usage users to do a litlle every now and then. The real share figure is then used to affect the user's overall priority.

The Real Effect

The Share Scheduler has no effect whatever on an under utilised machine, and in pratcice this means less than 45 users on our VAX 11/780. However, over this, the effects are very satisfactory. To be most useful, there must be, at any one time, two or more classes of user with different priority of access to the resources. For instance, during class hours, students are deemed to have greater rights to a reasonable response than, say, academic staff. This is achieved by an (heuristic) allocation of shares to each class of user based on expected average use of the machine, and bearing in mind the decay rate of the usage. For example, at Basser we allocate 10 shares to first year students (who are expected to use 3 terminal hours per week), 50 shares to academic staff, and 200 shares to staff programmers. This effectively bars academic staff from using the machine during periods of heavy usage as their figure is always fairly high, whereas first year students (whose usage decays substantially between their terminal sessions) can obtain reasonable response even during times of peak loading. On the other hand, the 200 shares allocated to programmers allows them to carry out normal system support programming with reasonable response at all times. All this is in addition to the fair distributon of resources within each class of user, as users who use the machine heavily are not allowed to reduce the response of similarly classed people who are working lightly.

With 45 users experiencing good response, it would appear that each user needs around 2.2% of a VAX 11/780, and it would be reasonable to nominate a share figure, say 0.5%, below which the system should actively discourage users from logging in, although we haven't done this. The consequence of not doing this, is that if a user's share drops below say 0.1% during a session when many more people have logged in, he may get "marooned" with insufficiant resources to allow log off (there being no cpu allocation enough to execute the "exit" system call from the

shell!).

A user may discover his share during a terminal session, and watch it change as costs are incurred, and he is of course informed of it at login. However the changes can be quite dramatic if for instance a user is the first of 75 to log in after a system reboot. It might be advantageous to inform a user whose share has dropped below a magic figure during a session, so that he has a reasonable chance to log off and let someone else use the terminal to greater advantage.

Tools

There are various system monitoring programs to display the performance of the scheduling algorithm, and nearly all parameters can be changed dynamically. In particular all charges can be varied to reflect administative policies about system resource costs. At Basser we charge at a fixed rate between 9am and 5pm of a working day, 50% of that until 10pm, and 10% overnight. Also users using the "nice" command get charged less for cpu time as appropriate.

Implementation

The actual scheduler is one page of C code and uses floating point extensively, although Hume has proposed a model using integer arithmetic. On the Vax, floating point is easier as the code is much cleaner, and the floating point instructions are reliable. The C compiler produces extremely inefficient floating point code, nevertheless, running once every 4 seconds, the scheduler consumes less than 0.5% of the cpu time managing 80 users on the VAX. Recoding the algorithm in assembler would reduce this to around 0.2%.

Charges are levied throughout the system at the various "cost" points. This is done by adding a variable charge for the resource type (obtained from a charges array) into a per-user "cost" variable.

The scheduler is invoked by the clock interrupt routine requesting a software interrupt at a low priority, once every 4 seconds.

The algorithm first scans all the per-user structures building up an idea of total rates and shares, and simultaneously decaying the usage and rate figures after adding in the cost. The rate and usage totals are then biased by (tunable) factors reflecting their relative weights in the priority calculation.

A second scan of the per-user structures then calculates the per-user priority to be used by the low-level scheduler. A user's share of the machine is thus reflected in his priority relative to the priorities of all other users.

Example Monitor Output

The following is a typical output produced by one of the monitor programs used to tune the algorithm.

```
Wed Oct 22 16:49:37 1980

Lnodes: 78    Users: 76                    Averages:
Usage factor  Rate factor Total rate       rate      usage  shares
4.224907e-05 1.343931e-03    2172346      21609  8.373e+07 37.1053
Constants:-
syscall   bio   tio   tic click maxnice ufactor rfactor    usagek
   1360  1000   283  8333     1      18       1       2 0.9999834
 **
****
****
****
*****
*****
*****
*****
******             *
*******  *         *
******* **         *
*************   ***
0 2 4 6 8 0 2 4 6 8   background = 1
```

There are 76 users logged on, not including the idle process and the system user (root). The "Usage" and "Rate" factors are intermediate values used by the scheduling algorithm, the "Total rate" value includes that of the system and the idle process, whereas the averages are for "real" users only. A real user is one who has been allocated shares. The constants refer to the charges being made for each of the resources, "usagek" is the rate of decay applied to usage figures, effectively 20% per day.

The histogram shows the number of users vs. their priority.

Distributing C Compiles

Peter Gray
Department of Computer Science
University of Wollongong
pdg@cs.uow.edu.au

March, 1993

1 Introduction

The Department of Computer Science at the University of Wollongong operates a computing environment similar to most computer science departments worldwide. The department runs four server machines and a laboratory of diskless workstations. Users may access the server machines via X terminals or ascii terminals. All machines are connected to the campus wide IP based network.

The majority of the server workload comes from undergraduate teaching, comprising the usual edit/compile cycle. By far the most common language being used is ANSI C.

The main server machines routinely support 30 to 80 users with load averages between 1 and 10. Although the workstations are usually in use, their load is typically much less, typically less than 1.

In addition, load on the servers is highly variable. Long periods of medium load are followed by relatively short (of the order of minutes) periods of much higher load with a corresponding deterioration in response time. It was noted that during these periods of high load it was common to find several C or C++ compiles running simultaneously.

This paper describes a system written at Wollongong to distribute C compiles off the heavily loaded servers onto the lightly loaded workstations in such a way as to be totally transparent to the user. This is feasible because the compilers involved always read and write files and may not be used as filters.

2 Look before you Leap

Although it appeared likely that the periods of high server load were attributable to high levels of compiler usage it was thought that it would be advisable to monitor CPU usage by the C/C++ compilers before undertaking work based on the above assumption. To this end, a small front end to the compilers was written and installed to collect statistics on the CPU

resources used per compile and how many compiles were being executed per day. After a few weeks the results were analysed.

Since problems with load only occur in "prime time" (approximately 9:00AM to 5:00PM, Monday to Friday) data from outside these times was discarded. The results were surprising.

The average C compile consumed approximately 3–4 seconds of CPU time on our SUN 670 (40MHz SPARC). C++ compiles were even more expensive, usually consuming more than 5 seconds of CPU time.

The C compiler was invoked more than 1000 times per day in the region of "prime time", consuming approximately 1 hour of CPU time in the 8 hour period.

This was enough to convince me that offloading the compiles would be worthwhile. In addition, offloading compiles onto a lab full of "idle" workstations would allow fast parallel makes and drastically cut the time required to compile large software packages.

3 Starting environment

For such a system to work there are several starting requirements. Firstly, the workstations on which the compiler will actually run (the compile servers) must have the same CPU architecture as the machines from which you wish to offload compiles (the compile clients). In addition they should all be running the same version of the operating system and language libraries.

All the machines involved should share a common user list, ie a single password file. In our case we run a single password file distributed by SUN's Network Information Services (NIS).

The machines should all agree on the current time. This is important since systems like `make` depend on correct file timestamps.

In addition, file systems accessed by the compiler must be accessible from the compile server in a similar if not identical fashion to the compile client.

This last issue is perhaps the most important and the hardest to achieve. The approach we use is to use Network File System (NFS) and SUN's `automount` to create 2 "virtual" filesystems accessible in a common fashion from all machines. All user's home directories are accessed via the `/home` virtual filesystem and all locally installed software and libraries etc are accessed via the `/share` filesystem. In general, users never need to know where particular files are located and all access to files is through the virtual filesystems. This was originally done for two reasons.

Firstly, it allows the systems administration staff to move file systems around, even between machines, without having the users aware of any changes.

Secondly, it allows replication of nearly all the filesystems in the `/share` virtual filesystem giving increased performance and reliability in the face of system outages.

Figure 1 shows a possible relationship between the user's view of the filesystems and the

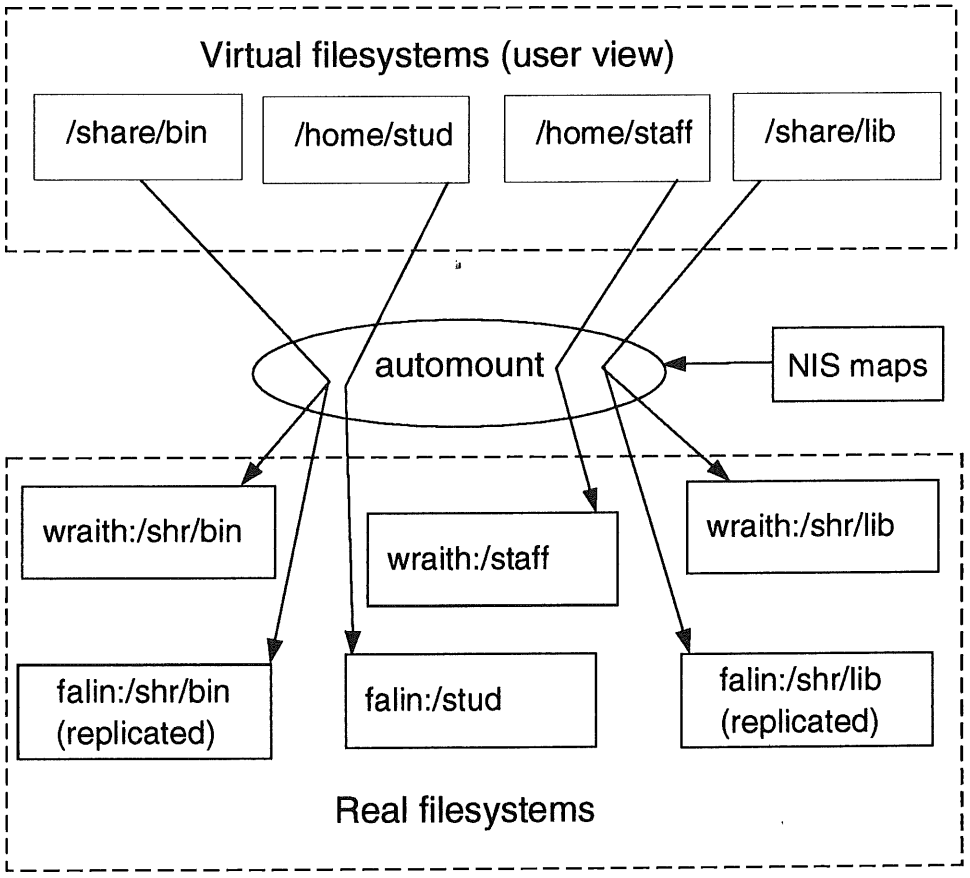


Figure 1: Virtual filesystems via automount

underlying physical filesystems.

The fact that the vast majority of file access performed by users takes place through virtual filesystem pathnames and that these pathnames are consistent across all machines greatly simplifies the problems of ensuring the remote compile process has access to the same files as the local user.

4 System components

The system is comprised of 5 components.

- A shell script which runs on each workstation to record the workstation's status. The status of a workstation is comprised of its load average and its free virtual memory. These parameters are used to determine if the workstation is a candidate to act as a compile server. This information is written into a file in the workstation's root filesystem.
- A shell script which runs on the boot server(s) which collects the status information and transfers it to the machine(s) acting as the compile server broker(s). See the next item.

- A compile server broker which is an RPC (Remote Procedure Call) server supporting a single RPC. The RPC returns the name of the next available workstation ready to act as a compile server. It returns NULL if there are no workstations available.
- A remote compile daemon, started on demand by `inetd` on the compile servers, which actually runs the real compiler.
- A local client to initiate and monitor the remote compile if possible or simply run the compiler locally if remote compilation is not possible.

4.1 Status scripts

Each workstation which may be a candidate to act as a compile server, runs a small shell script which repeatedly sleeps for some period (typically 60 seconds) then finds the machine's load and free virtual memory and writes the information to a file in /etc.

The boot server(s) for the workstations collect the information for each workstation and pass it to the machine(s) running the compile server broker. This script also deletes its input files as it reads them. This has the effect that if a workstation crashes or is switched off it quickly vanishes from the list of available machines.

4.2 Compile Server Broker

The compile server broker implements the server half of an RPC which returns the name of the next available workstation ready to act as a compile server. The broker process keeps a table of available workstations and their status in memory. When an RPC arrives, it checks the timestamp on the table against the current time, and if the table's information is considered obsolete (older than a build time constant) then the table is re-read from disk.

The table only contains the workstation's load and free virtual memory. For a workstation to be considered available, the load must be less than a threshold value and the free virtual memory greater than a similarly defined threshold value. The above thresholds are set at build time.

Each time a workstation's name is passed out to an RPC client, its load is incremented and its free memory decremented by configurable constants. This prevents a workstation being asked to run more than a few compiles per minute.

The decision to control most aspects of the system with compile time constants rather than configuration files was taken deliberately to ensure the system worked as quickly as possible.

The RPC server is called on demand and waits for 60 seconds of inactivity before terminating. In normal operation it tends to run all day without being restarted.

A useful enhancement would be to use UDP rather than TCP as the transport protocol for the RPC. This would reduce the overhead of the RPC call.

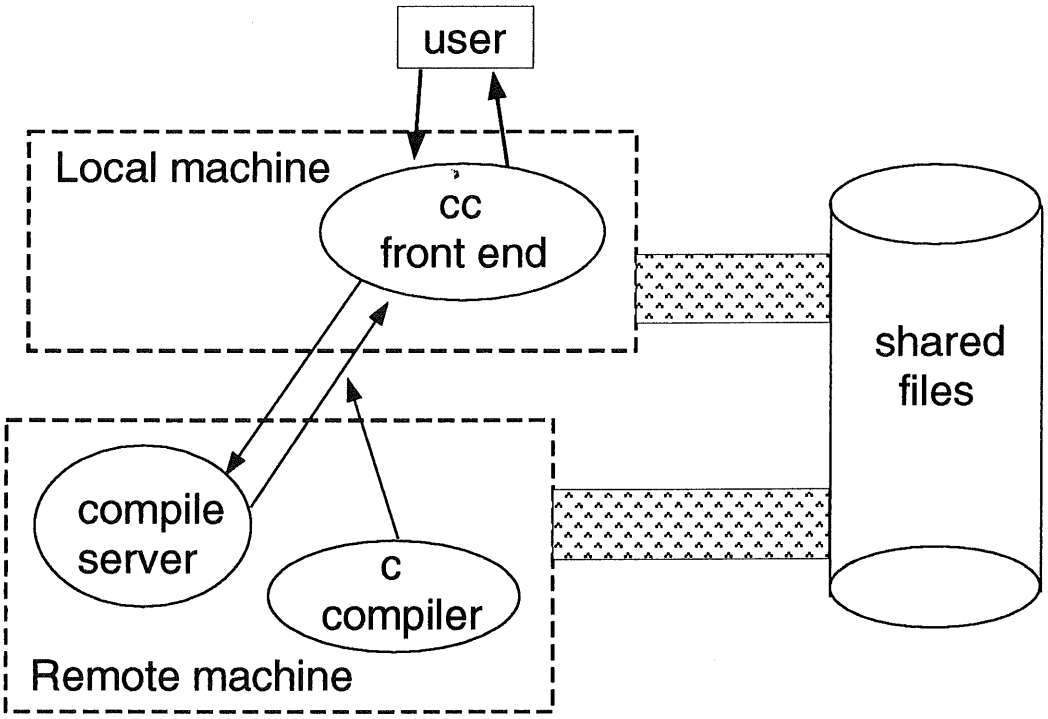


Figure 2: Process Layout during compilation

4.3 Remote Compile Daemon

The remote compile daemon is started on demand on a workstation to start and oversee the actual compile. It implements a simple protocol to talk to the compile client. This is discussed later. It also has to perform authentication checks.

4.4 Local C Front End

The local C client is meant to be installed as the file users execute to invoke the compiler. It determines if the compile is a candidate for being run remotely and if so, initiates contact with a remote compile server and monitors the compile until completion.

Figure 2 shows how the local compiler client and the remote compile server interact during the compile.

5 Local Compiler Replacement

The major task of the local cc/CC replacement is to determine if the compile can safely be run remotely. If it is a candidate to run remotely it is a fairly straightforward procedure to do so. If the compile can not be run remotely the local client simply executes the actual compiler with the argument list specified.

The only real requirement for the compile to be able to run remotely is that the compiler not attempt to reference files that are not accessible through the virtual filesystems. However, it is difficult to determine this easily.

For example, it is tempting to think that if the current directory is accessible via the virtual filesystems then all relative pathnames are also accessible. However, symbolic links as well as .. entries in pathnames can make the above assumption false.

To do the job properly would require the client to completely parse the argument list extracting out all filenames. These filenames would then have to be examined (via `stat(2)` or similar) to determine if they resided on the same physical filesystem as the current directory or if they resided on other filesystems that were accessible via the virtual filesystems.

The technique actually employed is in no way rigorous. It is however fast and gets it right almost without exception. The client simply examines each argument in turn and compares the start of the argument string against a list of inbuilt strings set by the installer.

The client uses the name it was called as (argv[0]) not only to determine which compiler to invoke but also to obtain the list of strings used to examine the argument list. The code fragment below indicates how the structures are defined and initialised.

```
struct c_info {
        char *basename;         /* my invocation name */
        char *compiler;         /* the real compiler */
        char *bad_options;      /* list of bad options or filenames */
};

struct c_info name_arg_list[ ] = {
        { "cc",   "/share/lang/cc",  "/ .. -o -I -L -temp= -Q" },
        { "acc",  "/share/lang/acc", "/ .. -o -I -L -temp= -Q" },
        { "dcc",  "/share/lang/cc",  "/ .. -o -I -L -temp= -Q" },
        { "dacc", "/share/lang/acc", "/ .. -o -I -L -temp= -Q" },
        { "CC",   "/share/lang/CC",  "/ .. -o -I -L -temp= -Q" },
        { "DCC",  "/share/lang/CC",  "/ .. -o -I -L -temp= -Q" }
};
```

It is obvious the above technique does not work in all cases. It could probably be improved by adding the ability to match argument strings against regular expressions rather than simple strings. This may be included in a later incarnation of the product. Despite its shortcomings the technique has yet to fail in everyday use.

If the above examination of the argument list does not preclude remote compilation, the client must then determine if the current directory is accessible through the virtual filesystems.

`getwd(3)` is called to determine the current directory name on the local machine. The local machine name, along with the local current directory name are used as keys to search a map. The contents of a sample map are shown below.

```
/*
 * This maps machine name, filesystem pairs into virtual filesystem
 * names. If the current working
 * directory starts with the string in the filesystem field
 * then the virtual_filesystem used is the current working
 * directory with the filesystem string replaced by the
 * virtual_filesystem string. If the hostname string is
 * empty, it matches all hosts.
 *
 * Example. machine name = draci
 *          current directory = /local/src/emacs
 *
 *          the above would result in the current directory
 *          being accessible through the virtual filesystem as
 *
 *          /net/draci/local/src/emacs
 */

        struct f_info {
                char    *machine;
                char    *filesystem;
                char    *virtual_filesystem;
        };

        struct f_info filesystem_map[ ] = {
                { "wraith", "/stud",            "/home/stud"},
                { "wraith", "/staff",           "/home/staff"},
                { "wraith", "/post",            "/home/post"},
                { "wyvern", "/shr/src",         "/share/src"},
                { "draci",  "/local/src",       "/net/draci/local/src"},
                { "draci",  "/local/users",     "/home/draci"},
                { "",       "/tmp_mnt/",        "/"},
                { "",       "/net/",            "/net/"}
        };
```

The comments describe how the map is used to convert a local directory name into the name of the same directory as seen through the virtual filesystem.

If all the above tests indicate the compile can be run remotely, the client executes the RPC to the workstation server broker and if a machine is returned the client attempts to start the remote compile server on the specified machine.

If any of the above steps fail the compile is run locally.

The decision to use inbuilt maps initialised at compile time rather than configuration

files or NIS maps was a deliberate attempt to maximise the speed of the client process in determining the feasibility of remote compilation.

The client determines the **umask** value the user has in place and also extracts from the process environment the values of certain environment variables, the names of which are established at installation time. The values of the user's **umask** as well as the values of the environment variables will be passed to the remote compile server in order to have the remote compile run in a similar and hopefully functionally identical environment to the one the user has established locally.

The client traps signals that could normally be generated from the user's keyboard and attempts to operate in an identical way to the actual compiler. An exception is the SIGTSTP signal which causes the local client to print a message saying that compiles may not be suspended.

6 Remote Compile Server

The remote compile server is started by **inetd** on a target workstation to start and oversee the compile. It communicates with the client via a simple, text based protocol discussed in the next section.

The server determines the identity of the user and checks that access is permitted (see the later section on security).

It then "assumes" the user's identity and attempts a **chdir** to the required directory, which is passed in from the client. If the change of current directory fails it notifies the client and exits.

If the change of current directory is successful, the server reads a command string from the client. This string is a Bourne shell command the server will pass to **/bin/sh**. The server then forks and the child execs **/bin/sh** with the command string passed from the client.

The only task of the server is then to wait for the compiler to exit while processing any requests from the client (see the protocol section). When the compiler terminates, it then returns the exit status of the compiler back to the client.

7 Protocol

The client and server process communicate via a protocol made up of the following commands and responses. Each element of the protocol consists of a line of text, the first character of which defines the protocol command or response. The remainder of the line may be empty or contain additional information relevant to the protocol element.

- POLL The client periodically polls the server to ensure the compile is proceeding normally. If more than an installation time defined number of polls become outstanding

the client assumes network or remote workstation failure. It then attempts to terminate the remote compile and run the compile locally.

- POLL_ACK The server acknowledges each poll from the client.
- CD The client sends one such command to the server per remote compilation. The command includes the name of the target directory.
- CD_ACK The server responds to a CD command with the status of the `chdir` system call.
- ABORT The client sends this command to the server when it wishes to immediately abort the compilation, for example, if the user generates a SIGHUP signal to the client. The server kills the compiler and terminates.
- STATUS The server uses this element of the protocol to indicate the remote compiler has terminated. The exit status of the compiler is returned.
- CMD The client sends one such command to the server per compilation. The command includes the string to be passed to `/bin/sh` to actually run the compile on the remote machine.

As figure 2 shows, output from the compiler is sent to the local client on the same file descriptor as is used to communicate with the remote compile server. The local client echoes output from the compiler to `stdout` while messages from the remote compile server (protocol messages) are processed silently.

The local client uses the fact that protocol messages all start with an ascii control character, while output from the compiler does not, to differentiate the two types of messages it receives. This assumes it is impossible for the actual compiler to generate lines with an ascii control character as the first character.

An obvious improvement would be to establish separate virtual circuits for the protocol and the compiler output. One reason for not doing this is the extra start up overhead this would create. Further work is required to determine if separate channels are warranted.

8 Authentication

The remote compile server is started on demand by `inetd` and runs as root. Before allowing connection to proceed it must first verify the user is authorised to use the machine. The identity of the connecting machine is available via the standard library function `getpeername(3)`. The server also requires the user name of the person attempting to connect.

This is obtained by utilising the RFC931 protocol which allows a machine to request information about the user name owning the process on the end of a TCP connection. A public domain implementation of a daemon to handle RFC931 queries is available as `pidentd`. Use of the RFC931 protocol for user identification means the user name does not have to be passed as part of the server/client protocol and greatly increases security.

Once the server knows the hostname and user name attempting to connect, it uses the standard library routine `ruserok(3)` to determine if connection is allowed.

9 Performance

The initialisation of the remote compile adds approximately 2 seconds to the elapsed time of a compile if compared with the time to compile on an idle server. Obviously the elapsed time for the remote compile is less than for a local compile if the local server is busy.

The system does have the effect of increasing network traffic since the files (including the executable of the compiler) will tend to be transferred over the network more so than if the compile ran locally. Of course, the above is only true if the files being accessed by the compiler are local to the server. Since our ethernets are not overloaded this does not cause us any concern. The net result is to trade CPU cycles for network traffic. Sites with overloaded ethernets may decide that such a trade off would not be a benefit.

The system works quite well. It was installed as the "standard" C compiler without users noticing any difference except that compiles now ran faster. In conjunction with a parallel make facility such as `gmake` it is possible to speed up compilations of packages by a factor of 10 or more by utilising an entire laboratory of workstations.

10 Conclusion

The concept of offloading compiles onto lightly loaded workstations from busy server machines seems to work well. The system may need some improvements as problems become evident.

Ensuring users have a logical view of the filesystems which is identical across all machines makes such systems much easier to develop.

11 Acknowledgement

This work was inspired by a paper given by Chris Vance at the 1992 Summer AUUG conference.

Homebrew Network Monitoring: A Prelude to Network Management*

Mike Schulze, George Benko and Craig Farrell
Department of Computer Science
Curtin University of Technology
Perth, Western Australia
mike@cs.curtin.edu.au, rete@cs.curtin.edu.au, craig@cs.curtin.edu.au

March 12, 1993

Abstract

A wide variety of public domain and commercial tools exist to help network administrators manage networks. Few of these tools achieve a satisfactory price/performance ratio for organisations with small budgets. To date, we have implemented a number of tools which allow us to examine and visualise network communications with a simple, intuitive, X based graphical user interface. These tools acquire knowledge on network communications via passive network monitoring, with a minimal amount of user intervention. This allows users who do not have an in-depth understanding of network architectures to gain an immediate, intuitive understanding of their network and its performance.

1 Introduction

With the sudden growth of UNIX™ based networks, a need for effective network diagnostic tools has arisen. Indeed, many tools have been implemented to help solve this problem, but are beyond the financial reach of network managers within small organisations.

Management of a small to medium sized network need not be an expensive, complex undertaking. Some of the functions provided by specialized network management packages can be implemented by a sub-set of publicly available tools. Of these tools, some are simple load monitors, others, such as *etherfind*[33] and *Tcpdump*[16] provide in-depth textual descriptions of local network traffic. These utilities may be sufficient for an experienced user, but can become cumbersome and/or uninformative in the hands of the uninitiated.

In this project, we attempt to extend the goals of these utilities by visualising network data. This has been achieved by applying a graphical model to a collection of continuously updating network statistics. These statistics are gathered by promiscuously monitoring the local network.

In the early stages of development, a number of tools emerged which collectively form the "Netman" project. The primary goal of the project is to provide an intuitive representation of network activity using graphical techniques. Several of the Netman tools have been inspired by non-text based network monitors like *EtherView*[14], *NetVisualyzer*[29], and *traffic*[36]. It is envisaged that Netman tools will supplement information provided by other monitoring tools rather than replace them.

The development of this project was entirely under UNIX™: we have not considered other operating systems[1]. At present we have implemented our work on Sun SPARCstation™ and DECstation™ 5000 series architectures. These platforms will be referred to as monitoring stations in the remainder of this paper. We chose to implement our work on these workstations for reasons similar to those outlined in [24].

All of our efforts are concentrated on monitoring Ethernet™[22] networks. A typical university ethernet, such as ours, provides us with a good environment in which to develop our project as it exhibits a wide variety of protocols.

There are two methods of passive network monitoring: real time analysis and retrospective analysis[2]. Our initial work concentrated on real-time traffic analysis, which forms the bulk of this paper. To a lesser extent, we explore an implementation of retrospective analysis in the form of a protocol analyser.

---

1. This is not to say that network traffic generated by hosts not running a UNIX derivative will be ignored.

*. This is a preprint of a paper to be presented at the SANS II World Conference on System Administration, Networking, and Security, April 18-23, 1993 Arlington, Virginia.

2 Statistics via Passive Monitoring

Passive network monitoring can provide a good insight to common network configuration problems. Examining the "wire" can often reveal problems such as protocol mismatches, incorrect routes, broadcast storms etc. We chose the passive approach because we were interested in monitoring normal network operations without interference from either a custom or established monitoring/management[2] protocol. That is, the act of observing a network should not directly interfere with or add to network activity.

Having access to local network data allows us the freedom to provide a multitude of statistical information. In this way, we are limited only by the amount of network data we can process "on the fly". A combination of specialised data structures and a reasonably reliable network tap forms the basis of the monitoring engine.

2.1 Packet Capture Mechanisms

User level access to physical network traffic requires an efficient mechanism which is able to interact with the monitoring station's network interface device drivers. Some modern implementations of UNIX have such a mechanism, e.g. Sun's *Network Interface Tap* (NIT)[34] in SunOS™, *Snoop*[30] in IRIX™, and the Ultrix™ *Packetfilter*[9]. All of these implementations are derivatives of the original "packet filter" developed at Carnegie-Mellon University in 1980[25]. At present we use the Ultrix packetfilter and NIT under SunOS. Although there has been a recent paper[19] outlining the ineffiencies of these mechanisms, they are standard system resources and provide sufficient performance and functionality for our work.

Data is captured by placing the network tap[19] into promiscuous mode and collecting statistics from the incoming packet headers. Information from each buffered queue, or "chunk" of packets, is decoded sequentially and distilled into the appropriate statistical data structure. No attempt has been made to make the packet decoding routines protocol independent; all protocol specific information is hard coded.

2.2 Determining Logical Network Connections

Finding connections between hosts forms the basis of our work. The method is simple: examine the source and destination fields of each passing packet. Once a source / destination address pair is obtained, it is stored in a continuously updated list of connected nodes. Each node which is identified as a source of network traffic, will have an associated list of connections to destination nodes. Both lists are updated as often as possible, in our case after each chunk of packets is read.

A node is never deleted from the list. Therefore the list is limited only by the number of hosts on the network and the system resources of the monitoring station. An idle[3] host is not removed from the list, instead it is flagged as idle. This way, if the host were to re-transmit, the idle flag could be toggled and the previously determined host statistics would still be available.

2.3 Data Flow Calculations

In order to extract useful information from network traffic, we have outlined a set of statistical categories which we consider important:

- per host source traffic flow
- per host protocol summaries
- per link traffic flow
- per link protocol summaries
- overall bandwidth consumption.

These calculations must be made each time a chunk is processed. This means that we must maintain a list of running totals each time a packet is decoded and then perform the calculations.

After each chunk is processed, a *chunktime*[9, 34] or time frame is established by finding the difference between the timestamps associated with the first and last packets in the chunk[4]. This value serves as a point of reference for all flow statistics generated from the running totals.

Per host source traffic totals are tallied by examining the source and length fields of each packet, then incrementing byte and packet totals for the corresponding node. These totals are then divided by the chunk time to yield a result which is stored in a sliding buffer of finite length. From this brief history of traffic statistics we can obtain a "sliding average" which is used to represent source traffic for that node. Link statistics are handled in a similar way, as running totals for link flows are also maintained.

Protocol summaries are more difficult: depending on the protocols considered most interesting to the user, each protocol layer must be decoded to the required level. For example, providing the frame type[8] of each Ethernet packet is not very useful if the network is predominantly

2. We are not attempting to criticise SNMP[4] in any way. The goal of the Netman project is to provide statistics on normal network operations as opposed to providing a management infrastructure. Although we realise SNMP MIBII[20] partially provides the monitoring functionality, it is lacking in per protocol based statistics.

3. The term "idle" is used loosely; the user must decide the parameters which class the host as idle.

4. This value is slightly inaccurate because we ignore the time gap between consecutive packets separated by two chunks. Although we realise that this value is not correct[24], it is sufficient for our purposes.

IP[28]. For this reason, we have introduced extra decoding routines that will provide summaries for UDP and TCP port connections. Although this involves more work for the monitoring station, we believe that the usefulness of this information outweighs the extra load.

Producing these summaries requires a moderately complex tree structure which is associated with each link[5]. The root of the tree is a pointer to a list of Ethernet protocols being used on a link with its associated byte/packet totals. Each frame type may have an associated sub-protocol, so in the case of IP, sub-protocols may include TCP, UDP etc. Following in this tradition, each sub-protocol may have a sub-sub-protocol, which we have termed the *port type*, solely for the TCP/IP suite. These statistics are running totals which are never converted to flow rates, but simply updated after each packet.

2.4 Limitations of Passive Monitoring

Restrictions to passive monitoring means the Netman tools can only access data on the local segment and other networks remain "invisible", i.e. cannot be monitored. Sun has attempted to solve this problem by writing a simple rpc service, *etherd*[32], which allows a monitoring station to connect to a host on another network and retrieve traffic statistics from the remote network. Unfortunately, our requirements were more complex than that of calculating a bandwidth usage statistic every second. We considered writing our own customised rpc service. However, this would have contradicted our requirement for passive monitoring, as such a service would inevitably generate a significant amount of additional traffic. Silicon Graphics takes a similar approach with *snoop*.

One of the limitations of passive monitoring is not being able to request information from the network. For example, SNMP will allow a management station to query a device (which is running an SNMP agent) to retrieve the requested MIB variables. All that can be obtained from a monitor are cumulative statistics concerning the active devices on the local network. This means that simple questions like, *is my host up*? cannot be answered if that host is idle[6].

Another inherent limitation on monitoring is that we are unable to detect physical level devices such as repeaters. That is, a fault occurring on a repeater segment, will only be detected via inability to reach hosts on that segment. Of course a traditional method can be used, such as sending an ICMP echo and waiting for a reply.

5. It is not the intention of this paper to provide a detailed account of the internal data structures we have implemented, so the discussion will be limited to a general overview of the statistical structures. Although it is useful to note each node has an associated list of links, and summaries for each node can be derived from this information.

6. Although *rwhod* will make a host "visible" if broadcasts are being monitored.

2.5 Effective Monitoring Strategies

The use of bridges can restrict the amount data which a monitoring station can access. A local Ethernet may be viewed by some as a single length of cable rather than a conglomeration of segments and bridges, but the use of a bridge[7] to fragment a network may serve as a "firewall" to the monitoring station (see Figure 1).

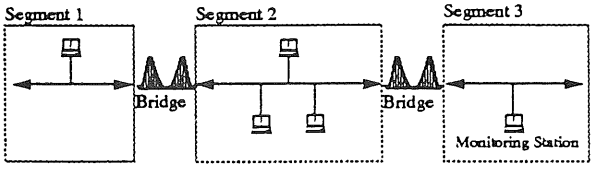


Figure 1. Using bridges to segment an Ethernet.

In this case the monitoring station will only see the traffic on or destined for segment 3. That is, even though the traffic on segments 1 and 2 may be considered local, the monitoring station can only see what is on the immediate segment. Traffic from segments 1 and 2 will appear to come from the ethernet interface of the bridge connected to segment 3.

Depending on the network activity that is considered most important, the monitoring station must be placed on a segment which is most likely to have the highest proportion of interesting traffic. For example, if remote IP traffic was of major concern, the monitoring station would be best placed on a centralised backbone which has a high exposure to foreign traffic.

2.6 Assumptions made on Physical Transmissions

Accessing data at the link level remains a problem, even if the workstation is capable of processing network data as fast as it arrives. The information being transmitted on the physical Ethernet will not always appear at the link level regardless of machine processing speed. The physical level hides collisions, checksums (valid or otherwise), corrupted frames etc. from the protocol level accessed.

Calculated load average, which represents the actual level of network traffic, must be as close as possible to the correct value. This means that certain assumptions about the data received are required. For example, a received packet is assumed to have had an associated preamble, frame check sequence, and an inter-frame gap[1]. This assumption compensates for the fact that these pieces of "lost" information are left at the physical level at the time of transmission (see Figure 2). Some network monitors fail to take this into account, creating a slightly inaccurate calculation of bandwidth usage, e.g., *xtr*[8][27].

7. Most modern bridges or level-2 routers[38] will only route packets if the destination Ethernet address is located on the other side of the bridge relative to the source device.

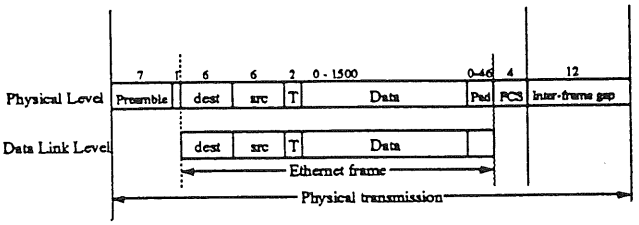


Figure 2. Frame lengths at different levels.

There is no facility within the interface "tap" to detect Ethernet collisions, therefore there is no way of determining the amount of information transmitted during a collision sequence[9]. This is yet another contributor to the overall problem of not being able to represent network traffic accurately.

Truncated or "runt" packets, that is packets with a length less than 60 bytes, should be treated as normal packets. This depends on whether the interface tap will pass the packets as valid transmissions. Since we have never encountered such a packet, we cannot confirm whether they are treated as a normal transmission. Misaligned packets have not been considered.

2.7 Packet Filtering

Packet filtering is the ability to focus on a specific protocol or a set of protocols, whilst the physical network interface is promiscuously capturing network data. Most network taps are able to filter packets at data link level and although Ultrix Packetfilter and NIT have this facility, we do not make use of it in our implementation. However, in order to avoid potential future problems when dealing with differences between filtering schemes across would-be platforms, and to provide access to all network traffic (used for calculating bandwidth statistics), we have implemented our own set of filters.

3 Visualising Network Data

One of the goals of this project was to provide tools with graphical representations of network traffic that are easy to understand, yet complex enough to retain all of the relevant information. Although it is hard to say if any single graphical representation could be useful in all situations [24], we believe a depiction exists for network communications which involve source/destination pairs. The remainder of this section outlines a model which is general enough to display overall network communications in a given network.

8. Unfortunately an assumption of 10 * 1024 * 1024 bits/sec was also made, which is not the correct speed of 10^7 bits/sec.
9. Although there is a way of determining the number of collisions on the interface since the machine was booted, this information is not of much use if a time-stamped collision indicator cannot be obtained.

3.1 Modelling Network Traffic

Our approach to traffic visualisation is to create a graph of network activity. To provide an accurate model of the network, the nodes, depicted as circles, represent devices, and edges, shown as line segments, represent logical connections between network devices. This graph is *connected*[37], that is any pair of nodes are directly reachable from one another.

Since typical networks have many devices sporadically but frequently communicating with each other, a representation of the network quickly evolves into a sprawl of lines and circles unless there is some form of logical node ordering and labelling. We chose to arrange the graph in a circular manner, with all nodes equidistant (this idea is similar to the approach taken by *NetVisualyzer* and *Etherview*). Each node has a unique label which identifies it from other nodes. The positioning of the label follows a perpendicular to the tangent of the network "circle" at the node position (see Figure 3).

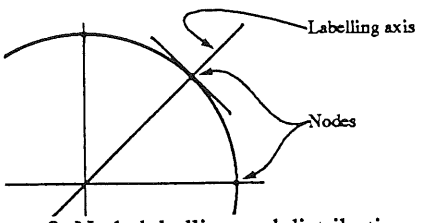


Figure 3. Node labelling and distribution.

A problem arises when we consider that each edge of the graph is undirected. That is, one host could send data destined for another, and the destination does not respond. Our model will create a connection between the two nodes but no indication of direction will result. Initially, we considered this a problem until we decided to identify the source and volume of traffic for each node.

Each node represents a network device which generates intermittent transmissions. The area of the node is a function of the source traffic generated by that device. That is, the source traffic for each device is directly proportional to the area of the circle, so devices generating a higher proportion of traffic will be more prominent.

Since each device may have multiple connections, line thickness may be used to represent the amount of traffic on a link between two hosts. Other packages, such as *EtherView*, display traffic statistics between hosts by a colour coding scheme. We have found this difficult to understand when the display update time was less than a second, and more than a few colours were used. *NetVisualyzer* uses colour intensity to represent traffic flow on a link, that is, a single colour is used and only the intensity of the link is varied. We believe our approach is more intuitive; if the link gets wider, more data is being transmitted.

The final aspect of our model involves the identification of *dominant* protocols on each link. A protocol is considered dominant if it has the greatest byte aggregate from all protocol transmissions for any given link. A pre-determined colour table assigns each protocol with a configurable colour code. Once the dominant protocol has been determined, the colour table is consulted and link colour applied.

So far we have assumed a network protocol that is bound to local communications only. If communications between hosts on differing networks are accessible[10], an augmentation of our model can be used to group hosts into a logical ordering of networks. This makes it possible to represent communications within network layer protocols, such as IP or DECnet.

We feel that the overall model just presented, is sufficient to provide a good picture of arbitrary network communications. We also feel that the model should not be expanded past its present form, for fear of introducing redundant complexities. For example, if arrows were used to imply direction of data flow on a given link, in most cases this information would be redundant as the majority of communications will involve a valid source / destination pair with bi-directional flow.

The question of whether or not certain statistics are relevant on a crowded display will always cast doubt on our claim that the model is general enough to show all network connections. In practise, we have taken measures to avoid screen overcrowding by implementing host/network time-out and compression mechanisms.

3.2 Animating the Model

Providing instantaneous feedback of network activities is an essential part of real time fault diagnosis. The original implementation of our model was conceived on a Silicon Graphics 4D/20 workstation using the native *gl*[31] library. This platform provided the necessary graphical resources and the network tap we required to produce a fast working implementation.

The *gl* library made it possible to develop a set of display routines which simply redraw the network, as often as possible. That is to, gather network statistics and redraw the network *ad infinitum*. This version was only capable of monitoring connections between network devices.

Once we were satisfied that the code had matured enough to warrant porting, it was decided to implement the code using the X11 Windowing system[23] to ensure graphical portability across platforms. The prospect of converting our carefully crafted display routines to *Xlib*[10] calls prompted us to search for an alternative method. After careful consideration we decided that we would use the *vogl*[13] library to integrate X with the *gl* calls we had grown fond of. Fortunately we had only made use of a few *gl* calls, all of which were supported by *vogl*.

3.3 Display Implementation

Bringing our ideas to life proved to be a balancing act between monitoring station resources and portability considerations. Our first implementation allowed much freedom when it came to animation, but lacked portability. Since we wanted to create a tool which contained our animated display as well as menus, push-buttons, graphs etc., we decided to explore the *X Toolkit*[21] to help solve our problem. Fortunately, we were able to implement a working version using *Xt*, *vogl* and the *hershey*[12] font libraries[11]. Initially we chose the *Athena* Widget set from MIT[26] as it is publicly available and provided the necessary Widgets we required. Currently, the Xaw3d[17] Widget set is used, purely for aesthetic reasons.

The present implementations of our work provide us with a set of robust, informative monitoring tools. Surprisingly, the display is dynamic enough to show network traffic in real time, without any significant lag between burst and display. Unfortunately this is not the case when the X display and the monitoring station are several networks apart.

4 Implementations

A need for informative, easy to use network monitoring tools within the international UNIX community was becoming more apparent. This lead us to experiment with public domain and system resident tools which have helped us to achieve what we have. Most of tools that we found are text based, e.g. *NNStat*[3], *Tcpdump*, *etherfind*, *nfswatch*[5] etc. and provided real-time statistics on network traffic. Although these tools have proved invaluable, we felt a graphical approach had far more to offer in terms of providing a user with a picture of all network traffic.

4.1 Monitoring All Ethernet Traffic: *etherman*

Our first implementation was directed at monitoring all ethernet host connections in real-time. Later, we added support for displaying the amount of source traffic from each node, and the amount of traffic on each host pair link. After testing *etherman* on different networks, it became obvious that we needed to be able to concentrate on "important" network activity by adding time-out and scaling functions. For example, a quiet network will appear

10. For example, Monitoring IP communications can often reveal source / destination pairs that do not reside on the local network.

11. The *Hershey* libraries were necessary because *vogl* does not support any form of text display. This also provided the font rotation which was available under *gl*.

static or uneventful, but it is possible to scale nodes and links to allow a better picture of relative network communications.

Network statistics are produced in both textual and graphical form. Dynamic statistics such as host and link traffic flow are displayed via our graphical model of network communications in real-time (see Figure 4). If a particular network phenomenon results during the course of execution, a postscript™ snapshot of the network is possible. Bandwidth consumption statistics are sampled once every second and displayed via a scrolling stripchart widget.

A textual dump of protocol summaries collected for each host and link is available. The information given is for the run-time of the program. These summaries are currently decoded for all Ethernet frame id's and all IP protocols. The output format for each host pair indicates the amount of data exchanged in bytes and packets, and the protocol in use.

We have used *etherman* to detect a variety of network problems and shortcomings. Because of its graphical nature, problems such as excessive bandwidth consumption and broadcast storms become easy to identify. It is also very useful for finding unexpected transmissions or unknown devices.

4.2 Focusing on IP Connectivity: *interman*

Using the augmented display model we can monitor a network level protocol, in this case IP, to display connectivity and dominant sub-protocols. In this way, networks are ordered in a circular manner much the same way as Ethernet devices in *etherman*. Unfortunately, this model will not display routing information as IP routing is transparent at this level.

As with *etherman*, we monitor and display traffic in real-time with update times varying because of traffic flow and monitoring station capabilities. As can be seen from Figure 5, different networks are shown as circles with hosts

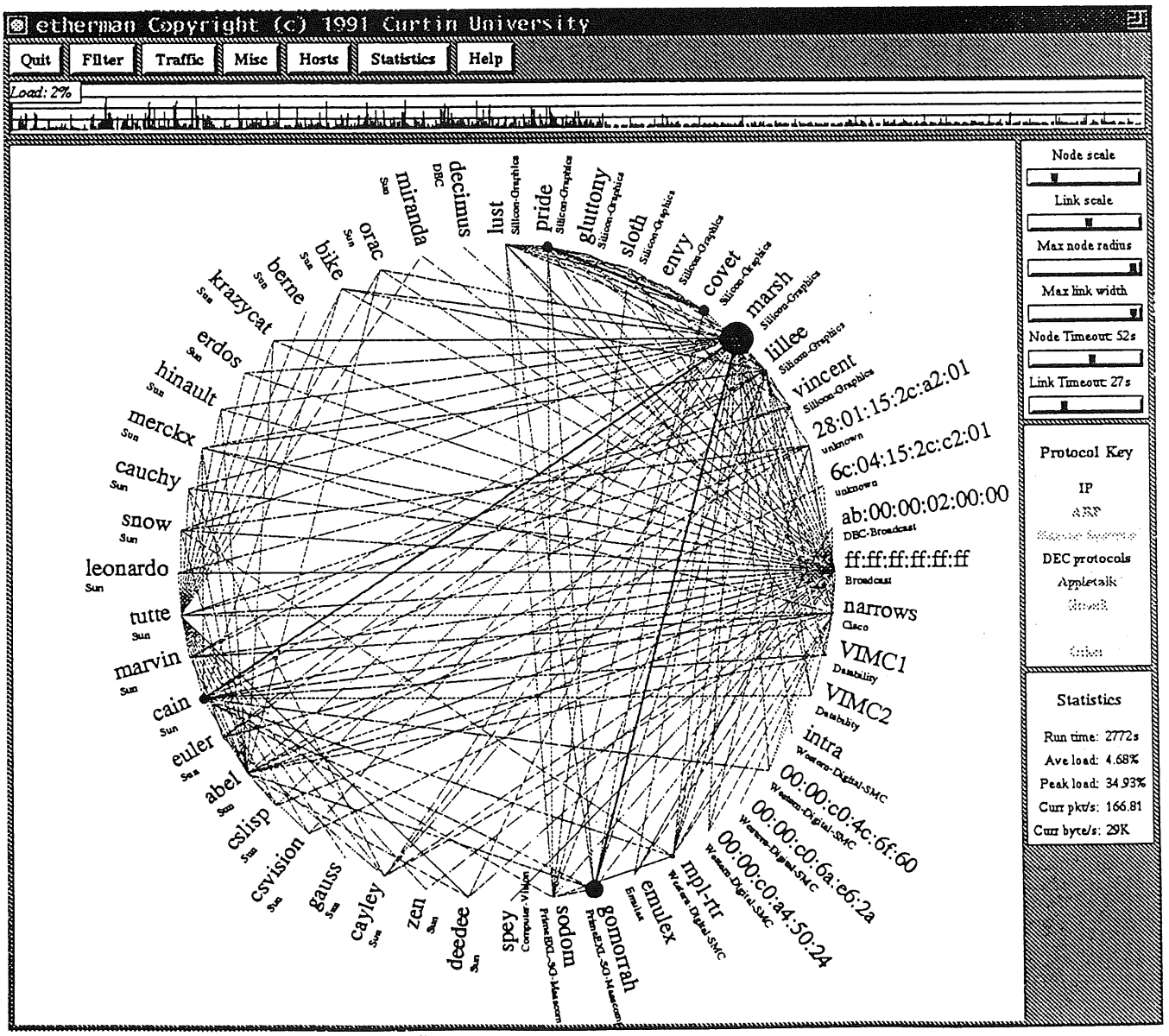


Figure 4. *etherman* monitoring all local traffic

local to each network listed around their respective circumferences. Hosts are connected between two networks via a single link; the colour of the link denoting the dominant protocol. Hosts and networks appear on the diagram as communications are monitored, and disappear when a host or network has been idle for a configurable period of time.

Figure 6 shows a sample summary of selected transmissions for host *cujo* on a per host, protocol breakdown. Since these summaries are derived from link statistics, we provide all communications involving the said host. This means that some information will become redundant when summaries are generated for each host involved in this example.

Several operations that extend beyond the scope of the original work have been included. These include *finger*, *telnet*, *ping*, network compression, *traceroute*, a modified *fping*, basic SNMP information, and a screen refresh. These options seem to contradict the notion of passive monitoring because they create traffic, but in practice these sorts of tools become useful when it becomes necessary to probe for network or host information. Other miscellaneous options such as protocol summaries, filtering, postscript dumps, and varying node and link time-outs are available via menu options and scroll bar adjustments.

```
Summary of Communications for host cujo.curtin.edu.au
    From cujo.curtin.edu.au to ubvmsb.cc.buffalo.edu
        TCP_SMTP pkts = 8 (0.56 K)
    From butler.acsu.buffalo.edu to cujo.curtin.edu.au
        UPD_DOMAIN pkts = 1 (0.08 K)
    From cujo.curtin.edu.au to uniwa.uwa.edu.au
        TCP_NNTP pkts = 489 (29.08 K)
    From uniwa.uwa.edu.au to cujo.curtin.edu.au
        TCP_NNTP pkts = 731 (349.04 K)
    From cujo.curtin.edu.au to wvnvaxa.wvnet.edu
        TCP_SMTP pkts = 9 (1.34 K)
        UPD_DOMAIN pkts = 1 (0.08 K)
    From cupid.curtin.edu.au to cujo.curtin.edu.au
        UPD_NTP pkts = 4 (0.35 K)
        TCP_NNTP pkts = 12 (0.73 K)
        TCP_LOGIN pkts = 98 (5.74 K)
    ...
```

Figure 6. Sample host protocol summary.

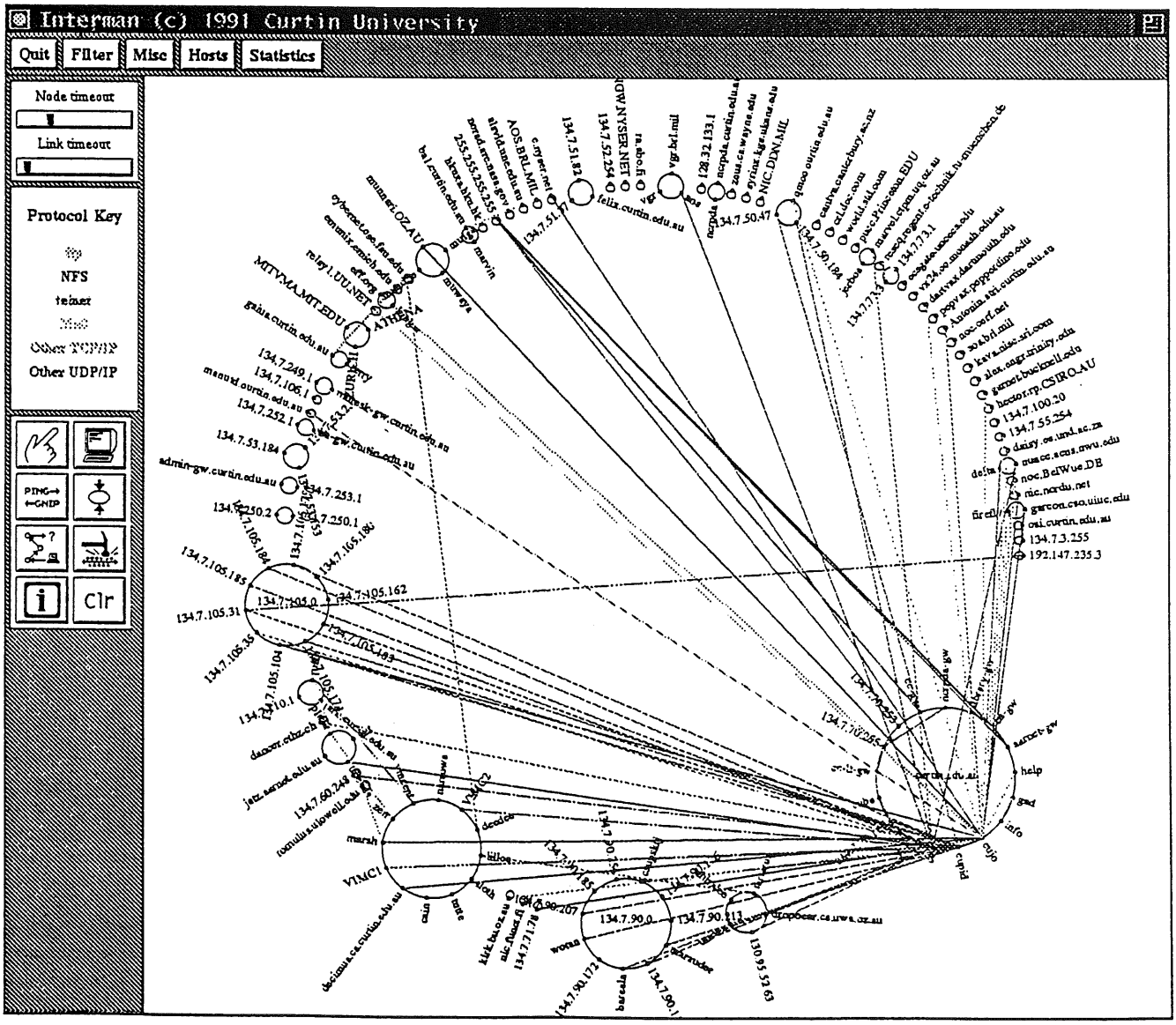


Figure 5. *interman* showing IP communications on the Curtin University campus backbone.

4.3 Packet Analysis; *packetman*

Branching away from the notion of real-time traffic analysis, we decided to implement a retrospective packet analyser. In practise, *packetman* can be used to decipher packet trains which are buffered, and optionally stored for future reference. A protocol analyser has the advantages of being able to decompose headers and examine protocol transactions in great depth. Our implementation, while far from complete, provides a good base from which we can work to provide comprehensive protocol analysis.

The display is segmented into three windows each providing a different view of captured network data. The top window is a sequential trace of captured data. Note that a filter may be applied before commencing a trace to allow the user to focus on transactions of interest. Each packet has an associated sequence number, timestamp, source/ destination pair, and a brief overview of the protocol contained within (see Figure 7).

A textual description of all decodable protocol fields within the selected packet is displayed in the middle window. At present, decomposition for Ethernet frame types, selected IP/UDP and IP/TCP, ARP and ICMP protocols have been implemented. The lower window gives a simple hexadecimal and ASCII dump of the entire packet.

5 Future Directions

5.1 Enhancements to Existing Tools

With the addition of extra data structures and protocol analysis routines, we have found that a reduction in performance has resulted. Fortunately, we have several options available to help improve overall display performance and statistical accuracy.

A recent paper[19] has outlined a new packet capturing mechanism, BPF[18], that claims a performance

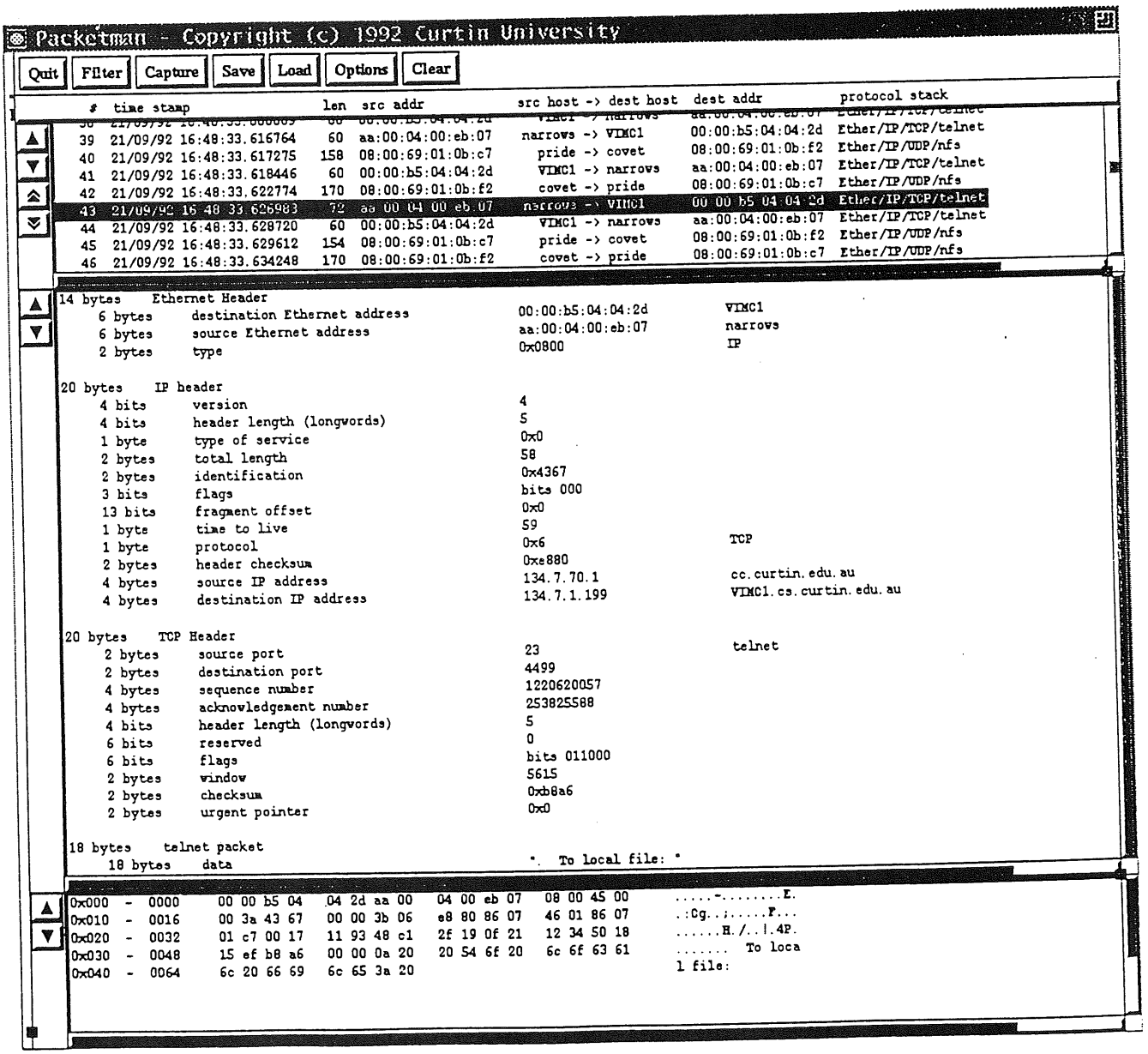


Figure 7. Using *packetman* to examine a telnet packet.

increase of 10 to 150 times over NIT. Although implementing code to support the BPF is trivial, we have no way of testing the implementations as we do not have access to the required SunOS kernel source. Unfortunately, the port of for Ultrix and other common platforms is not currently available, but it is hoped that a mechanism will be incorporated at compile time to determine whether BPF support is available, otherwise the system standard network tap will be used. This will allow us to support as many platforms as possible in the same manner as *Tcpdump*.

A major contributor to poor performance is the sheer volume of X protocol transmissions. Although this is hardly surprising within a dynamic graphical application such as ours, we can reduce the number of X server requests to a fraction of what they are now by improving font handling within the X server. Enhancements such as these[6] have already been made available in the public domain as extensions to the X11R5 font server. Although such extensions are currently available, they are not implemented on a broad cross-section of X capable workstations. We are hoping that extensions to the X protocol, or a similar mechanism, will be included as part of the X11R6 distribution to provide this support. This will allow us to re-work our display routines and provide a reduced display update interval.

Calculation of traffic flow statistics also has room for improvement. At present, our flow calculations are inaccurate due to inter-chunk gaps being ignored. We have decided to implement the *augmented loadring* algorithm[24] to provide us with the correct flow statistics.

Packetman requires a great deal of work in terms of additional protocol modules. We will continue to expand the capabilities of the IP module as best we can, but cannot guarantee that other major protocols such as DECnet, Novell, EtherTalk etc. will be attempted[12].

5.2 SNMP Management Station

Implementing a useful network management station using SNMP will be the most challenging aspect of the project. We have not decided how the display will work, as the question of the ideal method of auto-topology remains unanswered[40]. We would like to implement an RMON[39] capable management station that will access the active host information and connection matrices of RMON probes. To date we are still concentrating our efforts on passive monitoring, but hope to enter the management scenario in the near future.

5.3 Representing Physical Networks

A project was initiated early in 1992 to investigate the possibilities of physical network representations. It was envisaged that this tool would be used to document physical network schemas with a minimum of effort on the user's part. This tool would use icons extracted from familiar vendor logos, and be able to interconnect these icons with lines representing Ethernet cable and fibre optic cable.

Although a very successful undergraduate project, in terms of X programming and network design experience, this tool requires a great deal more development in order to be usable. We hope to continue the development of this tool and also perhaps to, couple it with a monitoring engine to provide a crude auto-discovery capability.

5.4 Geographical Traceroute

Most people are familiar with the internet route tracing facility, *traceroute*[15]. Some people may also be aware of the uumap project which attempts to identify sites globally. If these two information resources were coupled with the CIA World Bank II[7] world map, it would be possible to produce a "geographical traceroute" which displays the geographical "point-to-point" routes taken to reach a remote site.

The information provided by such a tool could be used as either a diagnostic tool or as a educational aid for those who have not grasped the basics of IP routing. Other possibilities such as overlaying national and international backbones and satellite links could provide a more in-depth view of Internet connectivity. This project is currently under development.

6 Conclusion

We have presented an overview of a research project currently underway at Curtin University. At present we have implemented some utilities which we feel provide the network administrator with an immediate and intuitive understanding of network utilisation. We have successfully used these tools to diagnose and correct a variety of faults on several networks.

These tools were not designed to replace existing network management systems, rather to supplement them. We believe that by adding the information provided by these monitoring tools to the facilities provided by management tools like *SunNet Manager*[35] or *HP Open View*[11], the network administrator will receive a more complete and powerful management system.

7 Availability

Interman, *etherman*, and *packetman* are available via anonymous ftp from host `ftp.cs.curtin.edu.au` as a binary distribution for DECstations and SUN SPARCstations. Depending on which machine type you require, the directories `pub/netman/sun4c` and `dec-`

12. If there is sufficient interest, we will consider releasing the source for public review.

mips contain the files: etherman.tar.Z, interman.tar.Z, and packetman.tar.Z. At present, source code is not available.

8 Acknowledgements

Much of our work has been a distillation of some of the publicly available source code provided by the Internet community. Many thanks to Jeff Mogul, Steve McCanne, Van Jacobsen, Craig Leres, Dave Curry, Robert Braden and Annette DeSchon and a multitude of developers across the Internet for their contributions of utilities such as *Tcpdump*, *nfswatch*, *NNStat* etc. which helped inspire the project. Thanks to George Ferguson for his efforts in *xarchie* which guided the development of our GUIs. Special thanks go to Phil Dench for his original implementation of *etherman* under IRIX (which will one day return), and his willingness to help when *gl* was all too hard. Thanks also to Peter Elford and Geoff Huston for encouragement, testing and colour postscript dumps. To all the current users across the Internet community, thanks for bug reports, suggestions, kind words and your patience. Last of all, thanks to all members of staff within the department who contributed to this project in many ways.

References

[1] Boggs, D.R., Mogul, J.C., Kent, C.A. Measured Capacity of an Ethernet: Myths and Reality. In Proceedings of SIGCOMM '88 (Stanford, CA, Aug. 1988), ACM.

[2] Braden, R.T. A Pseudo-machine for Packet Monitoring and Statistics. In Proceedings of SIGCOMM '88 (Stanford, CA, Aug. 1988), ACM.

[3] Braden, R.T., DeSchon, A.L. *NNStat*: Internet Statistics Collection Package -- Introduction and Users Guide Release 2.3 edition, USC / Information Sciences Institute, Marina del Ray, CA, 1989.

[4] Case, J., Fedor, M., Schoffstall, M., Davin, J.A. Simple Network Management Protocol. Request for Comments #1157, Network Information Centre, SRI International, May 1990.

[5] Curry, D., Mogul, J. *nfswatch*(8l) manual page. Purdue University, IL, March 1993.

[6] Deininger, A., Meyers, N. Using the New Font Capabilities of HP-Donated Font server Enhancements. The X Resource, O'Reilly & Associates, Inc. CA, Issue 3, 1993

[7] Dellinger, J. CIA World Bank II map data and source code. Stanford University

[8] Digital Equipment Corporation. The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications (Version 1.0). Digital Equipment Corporation, Intel, Xerox, 1980.

[9] Digital Equipment Corporation. *packetfilter*(4), Ultrix V4.2 Manual.

[10] Gettys, J., Scheifler, R.W., Newman, R. *Xlib* C Language X Interface, X Version 11 Release 5. MIT X Consortium, MA, Aug.1991.

[11] Hewlett Packard. HP Openview reference manual.

[12] Hook, D.G. The *hershey*(3) manual page. Melbourne University, Melbourne, 1992.

[13] Hook, D.G. The *vogl*(4) manual page. Melbourne University, Melbourne, 1992

[14] Hull, C. The *EtherView*(1) manual page. University of Vermont, 1991.

[15] Jacobsen, V. The *traceroute*(8) manual page. Lawrence Berkeley Laboratory, Berkeley, CA, Feb. 1989.

[16] Jacobsen, V., Leres, C., and McCanne, S. The *Tcpdump*(1) manual page. Lawrence Berkeley Laboratory, Berkeley, CA, June 1989.

[17] Kiethley, K.S. Three-D Athena Widgets (*Xaw3d*) source code, Jet Propulsion Laboratories, NASA, Feb. 1993.

[18] McCanne, S. The BPF Manual page. Lawrence Berkeley Laboratory, Berkeley, CA, May 1991.

[19] McCanne, S., Jacobsen, V. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In 1993 Winter USENIX conference proceedings (San Diego, CA, Jan. 1993).

[20] McCloghrie, K., Rose, M. Management Information Base for Network Management of TCP/IP Based Internets: MIBII. Request for Comments #1213, Network Information Centre, SRI International, March 1991.

[21] McCormack, J., Asente, P., Swick, R.R. *X Toolkit* Intrinsics - C Language Interface, X Version 11 Release 5. Massachusetts Institute of Technology, MA, August 1991.

[22] Metcalf, R.M., Boggs, D.R. Ethernet: Distributed Packet Switching for Local Computer Networks. Communications of the ACM 19(7):395-404, July 1976.

[23] MIT X Consortium. X11 Windowing System Release 5. Massachusetts Institute of Technology, MA, Aug. 1991.

[24] Mogul, J. C. Efficient Use of Workstations for Passive Monitoring of Local Area Networks. In Proceedings of SIGCOMM '90 (Philadelphia, PA, Sept. 1990), ACM.

[25] Mogul, J. C., Rashid, R.F., and Accetta, M.J. The Packet Filter: An Efficient Mechanism for User-level Network Code. In Proceedings of 11th Symposium on Operating Systems Principles (Austin, TX, Nov. 1987), ACM, pp. 39-51

[26] Peterson, C.D. Athena Widget Set - C Language Interface, X Version 11, Release 5. Massachusetts Institute of Technology, MA, July 1991.

[27] Pochmara, J. *xtr* source code, Dec. 1989, Oregon Graduate Institute, Beaverton, OR

[28] Postel, J. Internet Protocol, Request for Comments #791, Network Information Centre, SRI International, September 1981.

[29] Silicon Graphics, Inc. NetVisualyzer reference manual, Silicon Graphics, Inc. Mountain View, CA.

[30] Silicon Graphics, Inc. *snoop*(6D) manual page. Silicon Graphics, Inc. Mountain View, CA, 1990.

[31] Silicon Graphics, Inc. Graphics Library reference manual, C edition (version 4.0). Silicon Graphics, Inc. Mountain View, CA, 1990.

[32] Sun Microsystems, Inc. *etherd*(8c); SunOS 4.1.1 Reference Manual. Mountain View, CA, Oct. 1990. Part Number: 800-5480-10.

[33] Sun Microsystems, Inc. *etherfind*(8c); SunOS 4.1.1 Reference Manual. Mountain View, CA, Oct. 1990. Part Number: 800-5480-10.

[34] Sun Microsystems, Inc. *NIT*(4P); SunOS 4.1.1 Reference Manual. Mountain View, CA, Oct. 1990. Part Number: 800-5480-10.

[35] Sun Microsystems, Inc. SunNet Manager 1.1 Installation and Users's Guide. Mountain View, CA, 1991.

[36] Sun Microsystems Inc. *traffic*(1c); SunOS 4.1.1 Reference Manual. Mountain View, CA, Oct. 1990. Part Number: 800-5480-10.

[37] Tremblay, J.P., Manohar, R. Discrete Mathematical Structures with Applications in Computer Science. McGraw-Hill 1975.

[38] Trewitt, G. M. Topological Analysis of Local-area Internetworks. In Proceedings of SIGCOMM '88 (Stanford, CA, Aug. 1988), ACM.

[39] Waldbusser, S. Remote Network Monitoring Management Information Base. Request for Comments #1271, Network Information Centre, SRI International, Nov. 1991.

[40] Waldbusser, S. Exposing the Myths about Autotopology. The Simple Times, C/o Dover Beach Consulting, Inc., CA, 1(1):5-6, March 1992.

Technical Issues with Object Databases

Nik Trevallyn-Jones

About The Author

Nik Trevallyn-Jones is a Technical Sales Support Representative for Object Design, Pty Ltd, Australia.

He can be contacted at :

C/- Object Design P/L, 330 Wattle St, Ultimo, Sydney, 2007
Ph : (018) 646 701
Fax : (02) 212 2766
email : oda@extro.ucc.su.oz.au

Nik Trevallyn-Jones has worked in the computer industry since 1981, when he started with Prime Computer as a hardware engineer. While at Prime, he worked in software support with the operating system, and database products. From 1986 to 1992, he worked for Apscore Int. involved in 4GL and system coding and design. For the past two years, he has worked with ObjectStore, an ODBMS. In 1992, he joined his present employer, Object Design Pty Ltd.

Abstract

Today's complex applications are relying on software design and programming facilities such as Object Orientation. These systems require a database substrate powerful and flexible enough to cope with the complexity of the data structures, fast enough to provide real-time performance, and an interface simple enough to be usable in such complex applications. The modern ODBMSs are meeting these criteria in most aspects. This paper describes their major abilities, and discusses issues in their implemenation.

Terminology

The term Object Oriented Database Management System is a new one to the industry, and as such, the specific features associated with a product bearing this description can vary. There are a number of associated terms currently in use, and an equal or greater number of designs and architectures to match. Currently, the following terms are in use to describe systems which are similar, but differ in some unspecified manner :

Object Oriented Database Management System - OODBMS
Object Database Management System - ODBMS
Object Server - OS (not to be confused with the Operating System)
Persistent Object Store - POS

This paper discusses features which are usually associated with Database Management Systems, but much of the architectural issues can apply equally to all of the above. However, in the interests of simplicity, the one term, ODBMS, will be used throughout. Only where distinctions are reasonably clear, and accepted will any distinction be made.

Brief History

Research into ODBMSs started in the late 70's. During the 80's this work was refined, and in the second half of the 80's commercial products were available on the market.

The ODBMS research had a number of influences, including Virtual Machines such as Smalltalk, and the requirements of complex design applications such as CAD. Early systems were single-user and slow.

The commercial systems available are mostly based on high-speed relational database engines, are multi-user, and yet can outperform RDBMSs by an order of magnitude in data retrieval. However a number of more specific architectures are being implemented, with corresponding improvements in performance, and/or features.

Basic Features

Current ODBMSs provide the basic features required of any DBMS :

- Storage and retrieval (persistence)
- Restart / Rccovery
- Lock management
- Transaction model
- Data Distribution
- Queries (different to RDBMS models!)

In addition, a number of other features are normal in ODBMSs, which are not found in RDBMSs :

- Long transactions
- Versioned data
- Data navigation

Finally, a number of "advanced" RDBMS features are not normally present in ODBMSs :

- Triggers
- SQL
- Forms builders / 4GLs

Interface - Language Embedding

Unlike RDBMSs, which rely on a standard "database" language, SQL, ODBMSs embed their interface directly into the programming language, eg C, or C++. This is made particularly attractive by the features of OO languages, such as Smalltalk, and C++ which support language extension.

With most ODBMSs, the schema language is the binding language (ie the programming language). This almost eliminates the "impedance mismatch".

ODBMS Motto : If you can do it in memory, we can do it on disk.

Note : Some variation exists. Some systems provide an OO schema language, which is different to the OO programming language. Some systems provide a simple API of subroutines, rather than a language binding. This is sometimes seen as one of the distinguishing features between an ODBMS, which has a language binding, and an OS or POS which doesn't.

Design Concepts

In the 70's, the major problem facing complex computing was one of program size. It was exceding memory. The two approaches to addressing this were :

- Application driven - overlays
  Inconvenient and slow.

- Substrate driver - Virtual Memory
  Convenient, fast, and now typically hardware assisted.

In the 90's, a very similar problem exists; data is larger than memory (even virtual memory). The same approaches exist :

- Application driven

  The application driven approach is the more common in current systems. Typically, the features of the OO binding language are utilised to provide much of the functionality transparently. Typical of this approach is the provision of a general persistent base class, from which other persistent classes can inherit. While these new classes transparently acquire the database abilities from their parent, some amount of their mechanics still require explicit program intervention. Eg "dirty()".

- Substrate driven

  The substrate driven approach allows for the greatest transparency. Any type of data can be made persistent, regardless of its inheritance. The sbstrate is also free to utilise whatever speedup features are available from the supporting substrates in the operating system - caching, locking, signalling etc.

Both approaches are evident in current products on the market today.

Persistence - the model

While the specific implementation of ODBMSs varies, there are currently only three persistence models being used, to the author's knowledge : (these terms are not yet official jargon):

- Inherited Persistence

  A base persistent class is provided, from which other classes must inherit if they are to be made persistent (stored in the database). This can be restrictive, in that classes that do not inherit their persistence cannot be stored (eg integers). This can lead to a duplication of the class hierarchy.

- Orthogonal Persistence

  This model specifies explicitly that any data type can be made persistent (stored in the database). Although various models are possible, the only one currently available is based on memory. In this model, a new type of memory, persistent memory, is provided. Any data allocated in persistent memory is automatically managed by the database. This provides a single-level view of memory.

- Generalised Persistence

  This model provides a database data modelling system seperate from the program data modelling system. This general database data modelling system can be mapped to various language systems. The difference between the Generalised and Orthogonal models lies in the modelling system. The orthogonal model makes persistent any data type, as expressed in the bound OO programming language. The generalised model manages persistent instances of any data type as expressed in the generalised data modelling language, and provides a mapping of this to the programming language. To make an existing data type persistent, it must first be described in the database's modelling system.

The implementation of the system is dependent, to some degree, on the model used.

Implementation Strategies

Current ODBMSs are implemented on a number of architectures :

- Existing database storage technology, eg high-speed relational engines.
  (typically Inherited Persistence Model)
  A number of successful products are implemented using non-specialised storage technology, yet still exhibit data retrieval performance an order of magnitude greater that current RDBMSs

- Generalised Storage
  (currently Generalised Persistence)
  This type of implementation is based on a highly connected data model, designed to directly support the typical OO data models.

- Virtual Memory Mapping
  (currently Orthogonal Persistence)
  This model stores the data on disk in exactly the same shape as it is represented in memory. This eliminates the need for converting data from its disk representation to its memory representation. Some further structure is normally required to manage this data. The only current implementation models this management on virtual memory.

Technical issues

The database interface decision creates two main groups, the explicit API approach, and the transparent, language binding approach. The API approach is the simpler, and is more familiar to existing programmers. A set of subroutines are provided for explicitly manipulating the database. There is no new technology involved in implementing this interface.

The transparent language binding approach requires one of two things :

- A programming laguage which is extensible
- A preprocessor which makes the language appear to be extensible

Since OO languages support extensibility, the usual approach is to bind to an OO language, and use its extensibility. A preprocessor may be involved additionally, or as an option. Using C++ as an example, a common technique is to overload the "->" and "*" operators to transparently retrieve data from the database. Since these operators are class specific, this approach is also class specific, hence the persistent base class model. However, since there are no standard operators for handling *all* changes to an object, this approach usually requires some explicit method to mark the object as modified, for example "dirty()". Similarly, locking is typically left to the programmer to explicitly control.

To make the model independent of the class, a type-independent method of managing data is required. Hence the memory model. Using C++ as the example again, the operators "new" and "delete" are overloaded, to provide persistent memory. This has the advantage that any data type can be made persistent, but introduces the issue of meta-data. As it turns out, meta-data access is a desirable thing, despite the OO pursists' view that it isn't.

A combination of the two approaches is also possible. A Generalised Persistence model can map its generalised data structures into the model of the bound OO language. This mapping handles all aspects of the actual persistent object's behaviour, including method invocation.

Pointers and such

The next challenge in a transparent language binding is address space management. Effectively, the database represents a larger address space than the virtual address space of the machine, so some kind of address mapping is required.

One common approach is to implement "smart pointers". This is a class that looks like a pointer to an object, but is in fact an object in its own right, that contains some kind of Object Identifier (OID). The smart pointer provides the address mapping from OID to the actual data. One problem with smart pointers is that most OO languages don't provide complete extensibility in the area of pointers. Hence smart pointers may be smart, but they're not pointers.

The only real alternative to this is to manage persistent objects with true pointers. This requires a true memory mapping architecture, so objects retrieved from the database can be placed in memory, and the

associated address returned to the application. This approach requires that the database software understands the virtual memory system far more intimately than does the smart pointer approach. However, the benefits are in true transparency, and in pointer dereference performance.

The differences between these models really becomes apparent when pointers are stored within persistent objects. OIDs are typically 96 bits in size, although this varies between 40 and 128. In a highly connected database, this can cause the database size to be two or more times the size one would expect, due to the difference between 32 bit real pointers, and (say) 96 bit OIDs. Additionally, the class hierarchy has to reflect the presence of the OIDs in place of pointers.

Storage and Retrieval

ODBMS systems implement a variety of mechanisms which combine to provide data retrievel performance an order of magnitude faster than current RDBMS systems.

- Object Clustering
  Optimises disk and network traffic, by providing for efficient pre-fetch. Since data is always moved around in some form of packet (disk page, network packet, etc) then clustering related data into the same packet means accessing one object in the packet brings all the objects in that packet into memory. Accessing the other objects in the packet involves minimal or no further overhead.

- Object Caching
  Keeps frequently accessed objects local to some process. Caches can be implemented in both memory and on disk, usually both, as appropriate. Caching optimises both disk and network traffic.

- Lock Caching
  Since objects can be cached, it follows that caching the locks would bring a further improvement in performance. Lock caching can optimise both server activity, and network traffic, in distributed environments. Adjusting the lock caching policy can adjust whether the optimisation is biased toward read-access, or write access.

- Object Navigation
  Pointers from one object to another are followed directly to the target object, by the database itself. This reduces the programmer effort in finding and managing large amounts of data, and allows the database system to optimise effectively.

The generic API model provides the programmer with standard functions to retrieve and store objects from and to the database. This is essentially the same as the interface provided by current RDBMS systems. However, the clustering and caching facilities still provide exceptional data retrieval performance.

The language binding model, in contrast, has contrived to allow normal object navigation to occur on persistent data. Following a pointer from one object to another results in an automatic retrieval of the object if it was previously unmapped. In the case of the persistent base class approach, the overloaded "->" operator performs this work. In the case of the memory model, the object is "page-faulted" into memory. In either case, the programmer is relieved of the concern of what data has already been retrieved from the database, and what has not. Pointers are simply navigated, and the data is there. The caching schemes contrive to ensure that data that was recently touched can be revisited with minimum overhead.

Object Mapping

One of the most interesting aspects of the ODBMSs is how they implement the actual mapping of a persistent object from disk into the program memory. The traditional approach has been to store the data on disk in some appropriate form, and connect it to a program object that manages the mapping. Both Inherited and Generalised persistence models can be implemented this way.

A more interesting approach is that taken by the Virtual Memory Mapping architecture. In this approach, objects are stored on disk in the same form as they are represented in memory. Retrieving data involves mapping it from disk into the process' address space. This is accomplished by using the page fault mechanism of the underlying Operating System. When a pointer to an unmapped address is dereferenced, a regular page-fault occurs. This is trapped by the Database, and the address examined. If the address is within the address range of a persistent segment (one that is managed by the database), and is currently un-mapped, then the database fetches the appropriate page / cluster / segment, depending on the current prefetch, and maps the data. The page fault continues, and the pointer dereference now refers to a valid address, and continues correctly. If the Operating System page-faults the data out, then it can page fault it back in without further work by the database.

A similar approach is used to detect object modification, and the mapping of changes back to the database server at transaction commit. Initially, when any page is mapped to memory, it is locked against update. Therefore, any attempt by the program to modify the contents of that page causes a signal. Again, the database traps this signal, documents that the page is modified, and unlocks the page. At transaction commit, all modified pages are flushed back to the database server.

To enable it to manage its own area of virtual memory, the database needs only to keep a simple map of segments, by address range. By this means, any memory address can be quickly examined, and its status as being either transient or persistent determined.

The object of this approach is performance. The mapping from disk to memory and back is very efficient, with minimum processing and memory overhead. The pointers between all objects are regular 32 bit pointers, so pointer dereference incurs minimal or no overhead once the object is mapped.

Lock management

As with RDBMSs, the performance / accessability compromise of locking granularity occurs with ODBMSs as well, although Long transactions address the issue to some degree.

- Object level and sub-object level locking
  Involves the usual tradeoff between the high accessability of data, with the performance overhead of frequent locking, and large locking tables. Many applications don't need locking at such a fine granularity.

- Page level locking
  Usually a higher performance option, since most objects are far smaller than a page. However the possibility of "spurious conflicts" arises. Object clustering can be used to avoid this. Page level locking implementations can utilise the hardware page lock speedup mechanisms.

- Cluster level locking
  Similar to page level locking, although the hardware speedups may not be available, however the cluster size can usually be adjusted.

ODBMSs don't currently support all the flavours of locking model found in RDBMSs. The trend is toward more being supported.

- One Writer OR many Readers
  Normal locking model, implemented by most ODBMSs

- One Writer AND many Readers
  Enhanced concurrency. Not commonly implemented by ODBMSs, although the trend is for more to implement it.

- Many Writer AND many Reader
  Often implemented with versioned data, and "long" transactions.

- No locking
  A number of ODBMSs allow the locking to be turned off.

Transaction model.

The transaction model effects not just the restart / recovery features of an ODBMS. Other issues include :

- Lock management
  Systems that support automatic locking acquire locks as required, and release them at the end of a transaction.

- Persistent address spacce management
  Processing data sets larger than the virtual address space (4Gb, with 32 bit architectures), the mapping of persistent objects to the virtual memory needs to be released, so more data can be mapped. This release and re-mapping should not occur except at transaction boundaries.

- Versioning
  Versioned data implements the concept of a long transaction, also known as a persistent transaction. Such a transaction can be help open for days or weeks. Locks and persistent address management must work correctly for long transactions as well.

Distribution

ODBMSs are designed for teamwork style applications where distribution is not optional. Typically, the network is one of workstations, with roughly equivalent processing power, as opposed to the workstation / central server network expected by RDBMSs.

Issues addressed by current ODBMS products in distribution include :

- Lock management
- Transaction management
- Recovery - 2 phase commit
- cache consistency
- concurrency management
- change notification

Queries

There is currently no Object-SQL standard. There is also a diminishing likelihood of there ever being one :

- Complex data unsuited to an arbitary language
- Graphical tools better

Current query abilities in the various ODBMSs vary greatly, from fairly extensive, to almost non-existent.

- Class extents
- server-based queries
- client-based queries
- indexing
- Very large queries (> virtual address space)

ODBMS specific features

ODBMSs have a number of features not found in RDBMSs. These features are therefore newer to the market, and less well understood both by manufacturers and users of ODBMSs

Long Transactions

This feature is a result of the CAD influence on ODBMS design. It allows greater data accessability and hence concurrency than standard short transactions. A Long transaction could last from minutes to months.

Versioned data

This feature is tied to the long transaction feature. It is like having RCS embedded in the database. This feature usually embodies "configurations" and "workspaces". Although both are almost standard jargon, the implementations vary widely. This is probably the most complex, and least complete of ODBMS development.

Data navigation

A well understood feature. Network and hierarchical databases have had these features for years. In addition, programs have navigated data for years as well. In this instance, ODBMSs are simply bringing database technology up to the abilities of the programming languages.

ODBMSs understand implicitly that the data they contain can be navigated. Pointers between objects are understood by the database. This contrast with RDBMSs, which don't understand navigation.

Features not found in ODBMSs

Triggers

Objects are inherently smart, and active. The basic need for triggers is almost non-existent with ODBMSs. Many situations where triggers were required, Objects do a better job anyway.

Triggers are associated with the database, not the actual data. Copy the data, but not the trigger definition, and the "smarts" vanish. Move objects from one database to another, and the smarts move as well

SQL

Queries

Since ODBMSs do not place restrictions on the data schema, a generalised query language is almost impossible (how do you query a map?). ODBMSs provide varying query support, but nothing analagous to SQL. Generalised graphical query tools are not far away though.

General ODBMS abilities

ODBMSs can be characterised by a number of their abilities :

- High performance data retrieval
- Natural handling of complex data types
- Natural ability to handle pointers, and to navigate data
- Typically embedded into the programming language
- Optimised for handling large data sets.

ODBMS application profile

- Large data sets
- Complex data model
- Large amount of data retrieval and reuse
- Variable transaction duration

Summary of current applications implemented on ODBMSs

- CAD/CAM etc (computer aided whatever)
- Modelling
- Software Engineering
- Dept of Defence
- Real-time (scada)
- Navigational (GIS, spatial querying)

An Update on UNIX-Related Standards Activities †

by Stephen R. Walli
Report Editor
USENIX Standards Watchdog Committee
<stephe@usenix.org>

Corwin's Razor

In December, I wrote at length about a couple of fundamental problems with the structure and process of defining the POSIX family of standards. POSIX will collapse under its own structure if not rescued soon was the premise.[1] It was motivated by the concern that we will lose the existing valuable model for portable applications programming, if POSIX continues along its current path. These concerns settled around test methods requirements placed on the working groups, and language independent specifications (LIS).

It provoked some people to do the net equivalent of "writing to their congressman," and the Email addresses at the end of the article received some mail. It also provoked some discussion on the net, which is good, because this stuff is important to you if you care about writing C, Ada, and Fortran programs that are as portable as possible across the widest possible set of architectures. Your viewpoint is important! Ultimately, it was a source of a lot of debate and discussion at the IEEE POSIX meetings in January, which is responsible for developing these source-code portability standards.

One of the driving arguments in these discussions was who is the customer for this work. This sentiment is best embodied by something said by Bill Corwin, the chair of POSIX.4 (Real-time extensions), which became:

Corwin's Razor: If they're not willing to put their money where their mouth is, they're not a customer.

Let's see where it got us.

Test Method Madness Part II

I expressed a lot of concern with the creation of test methods standards. These are standards containing test assertions, based on the test methodologies outlined in POSIX.3, which could act as the basis for a test suite. I was concerned about the setting up of a directly competing body of text, used to create test suites to measure conformance of implementations of base standards, before the base standard had even received widespread implementation.

After the last editorial hit the net, I discovered something that only fueled my concerns. The test methods that were balloted as part of the XOM (Object Management) API, and X.400 API (P1224 and P1224.1) received no comment in ballot. Changes made to the test methods were done by the technical editor during ballot to keep them synchronized as best possible with changes made to the base API text. The POSIX.17 (X.500 Directory Services) test methods received very few comments in ballot, in relation to the volume of comments on the base API.

All three of these test methods *standards* are about to go to the IEEE Standards Board in March for final approval. One might think that their test methods weren't read very carefully by the balloting group, which was concentrating on balloting the base API that it cared about. Would you want NIST to choose these standards as the specification of a conformance test suite?

All three of these documents had their test methods written for them by a couple of contractors in the employ of X/Open. X/Open wants these base specifications to get through the standardization process. The process demands there be test methods for the base documents to exit ballot. There are now test methods. At least X/Open was willing to put its money where its mouth is. POSIX.10 (Supercomputing Profile) has just completed its first ballot cycle, and no, they received no ballot comments on their test methods section either.

There is an example, however, of a test methods standard that is, IMHO, a well-constructed standard. POSIX.3.1 (IEEE Std 2003.1-1992) is the test methods standard for the POSIX.1 base functionality. What's different about it:

- It lags the original standard by four years, since POSIX.1 was originally approved in 1988 (IEEE Std 1003.1-1988).
- At least four implementations of real test suites fed its creation. (The NIST PCTS, Perennial's POSIX.1 test suite, X/Open's VSX, built by UniSoft, and Mindcraft's CTS.)

1. *;login:*, January/February 1993.

† This is a re-print from ;login, the USENIX Association Newsletter, Volume 18 Number 2

- People that wanted to have a standard for test methods to measure conformance to POSIX.1 got together (i.e., found the time and money) and did the hard work of defining, drafting, and balloting a document.

- The document was balloted separately and seriously by a group of people that seriously cared about the outcome of the standard (i.e., the implementors of the base POSIX.1 standard), as well as the people that cared about having a standard test suite.

Many working groups that have written test methods for their base functional definitions, even just partially, feel that their base specifications are stronger for the effort. They aren't sure whether or not they've written good test methods. They don't care. Their base specification is better. That's what they came together to do.

You can't legislate a standard. Especially from a group of volunteers. Just because one group of people want a standard "test suite," does not mean that the group that originally got together to define and draft and ballot a standard for some base functionality wants to do the extra work of defining, drafting, and balloting a test methods standard. Corwin's Razor.

So what happened? Most of the Steering Committee on Conformance Testing's (SCCT's) discussions agreed that the market will speak. If and when it cares about standards for test methods, people will find the money and energy.

A motion was made in the Sponsor Executive Committee (SEC) to remove the testing requirement from balloting base documents. It was modified to "suspend" the requirement. It passed. The SCCT will consider if there are alternatives to the current process, but until such time as they report back to the SEC, the requirements placed on the wrong group of people have been lifted.

It does not mean POSIX considers test methods standards to be bad things. POSIX.3.1 stands as an example of a well-developed test methods standard. It also does not mean POSIX doesn't care about building good documents. Quite the contrary. A tool exists, called "writing test methods," that working groups can use, when and where appropriate, to improve the clarity and preciseness of their base specification. It does mean that the POSIX standards working groups feel that people that want test methods standards should go to the effort of building them.

LIS – Again!

The scope of POSIX.1 says that it is a standard operating system interface to support applications portability at the source code level. It is to be used by systems implementors and application developers. This would tend to indicate it should be a readable document. It is the official "contract" with which an applications developer can call up their systems vendor and say "this is supposed to behave this way," or vice versa. Just like the ANSI C standard. Together, these two documents define an environment in which to design more portable applications, written in C. (The Ada based POSIX.5 has similar statements in its scope.)

I raised concerns over the structure and format of the programming-language-independent specifications method of defining POSIX standards. It takes what I believe is a useful single-book, single-context format, as used in the current C-based POSIX.1 and Ada-based POSIX.5, and makes it hard to read and harder to use by creating a two-book, two-context standard.

The LIS debate is far more political and emotional. ISO is involved. The ISO Working Group responsible for bringing IEEE POSIX documents forward as international standards, WG15, requires that they be brought forward as thick, semantic LI specifications with attendant thin, syntactic language bindings.

This requirement was agreed to by the U.S. member body to WG15, which passed it on to the IEEE working groups back in 1988, which also agreed to do it. Some argue that the United States is not fulfilling its obligations if they don't follow the LIS approach.

This is just plain wrong. The IEEE is the POSIX standards development organization. The IEEE is a "transnational" organization, open to all. While the IEEE POSIX working groups are predominantly attended by people living in the United States, a fair number of people hail from other locations, and the IEEE POSIX working groups have never not entertained a person's point of view. They really don't care where you live. The "U.S. member body" to WG15 is merely the administrative point between the IEEE and ISO WG15.

The IEEE then spent a lot of time and effort, first defining a methodology, then applying that methodology where they could. As with the test-methods tool, working groups discovered holes and errors in their base text as they reviewed it critically, asking the question, "How would I express this concept such that I could write the Ada equivalent of it?" Or Fortran. They discovered they can more clearly express some of the meaning in their base documents.

The people most able to do this work in the IEEE POSIX Working Group's experience were the people in POSIX.5 (Ada) and POSIX.9 (Fortran). They did the painful exercise of critically reading the text of C-based POSIX.1, and recasting it into the words and programming semantics/syntax of their own language.

The funny thing is, they did this without the benefit of a language-independent specification of POSIX.1. The only way that the POSIX.1/LIS could be created was for the IEEE Technical Committee on Operating Systems to open the purse strings and pay a contractor to write the first drafts of an LIS of POSIX.1 and its attendant C binding.

And these other language groups, which pay money to come to IEEE POSIX meetings to do the work of building POSIX standards in their own language (which they understand well), are concerned with the current POSIX LIS methodology being used and the format of the documents it produces. I think I hear Corwin's Razor being sharpened.

The POSIX.5 Ada Working Group built their version of POSIX.1 without the POSIX.1/LIS. They feel it would have been easier to build their document if one had existed, but they don't know what that LIS would have looked like. They don't particularly like the one that has been produced.

They further chose to ignore the ISO requirement of "thin" syntactic bindings, which don't reproduce the semantic description of the "thick" base LIS document. They wanted their standard to be self-contained and readable, such that programmers would only require the one book on their desk. Their gamble failed! ISO WG15 refused to accept their standard for international standardization.

But wait! ISO WG9, the ISO Ada language group wants to fast-track the IEEE POSIX.5 standard. They feel it is a good standard. So maybe their gamble didn't fail.

So who actually benefits by presenting the standard as an LIS? Language-bindings writers certainly. But the people who already care enough to participate seem to be doing so. It doesn't take a huge number of people to set up a working group. There were only about twelve in the Fortran group (POSIX.9).

So what happened at the SEC? A motion came forward to remove the requirement for the current LIS method of defining POSIX standards. It was massaged into something considerably more diplomatic, which passed, creating an ad hoc committee to investigate the problem in detail, and without particular restrictions, and report back at the April 1993 meetings.

This is a good thing. To change the direction of POSIX at this point is not a trivial task to be taken lightly, nor decided too quickly. There will be ramifications no matter what the outcome of the report from the ad hoc.

I chair the ad hoc committee. If I am going to stir up all this fuss, then I am going to be made accountable for it. I am interested in your thoughts or concerns, so by all means Email me.

There was another wonderful quote that came out at the January meetings, that essentially said:

Standards organizations that choose to make themselves irrelevant deserve what they get.

This was actually made in reference to a completely different problem, but I believe it is very appropriate here. If we make these standards unusable, they won't be used. We will lose the "contract" for a portable programming model between applications developers and systems implementors.

I am repeating the list of Email addresses from the end of "POSIX – Caving in Under Its Own Weight." I believe it is still important that you make your concerns known to the people that can actually make the decision about this.

Position	Name	E-mail
IEEE Concerns		
Chair SEC	Jim Isaak	isaak@decvax.dec.com
Vice Chair Interpretations	Andrew Twigger	att@root.co.uk
Vice Chair Balloting	Lorraine Kevra	l.kevra@att.com
Chair Steering Comm on Conf Testing	Roger Martin	rmartin@swe.ncsl.nist.gov
Chair Project Management Comm	Shane McCarron	s.mccarron@ui.org
Chair POSIX.1	Paul Rabin	rabin@osf.org
Chair POSIX.2	Hal Jespersen	hlj@posix.com
Chair POSIX.3	Lowell Johnson	3lgj@rsvl.unisys.com
Chair POSIX.4	Bill Corwin	wmc@littlei.intel.com
Chair POSIX.5	Jim Lonjers	lonjers@prc.unisys.com
Chair POSIX.6	Dennis Steinauer	dsteinauer@nist.gov
Chair POSIX.7	Martin Kirk	m.kirk@xopen.co.uk
Chair POSIX.8	Jason Zions	jason@cnd.hp.com
Chair POSIX.9	Michael Hannah	mjhanna@sandia.gov
Chair POSIX.12	Bob Durst	durst@mitre.org
USENIX Institutional Rep	Jeff Haemer	jsh@usenix.org
EurOpen IR	Stephen Walli	stephe@mks.com
DECUS IR	Loren Buhle	buhle@xrt.upenn.edu
OSF IR	John Morris	jsm@osf.org
UNIX International IR	Shane McCarron	s.mccarron@ui.org
X/Open IR	Derek Kaufman	d.kaufman@xopen.co.uk
WG15 Concerns		
Convener WG15	Jim Isaak	isaak@decvax.dec.com
US Head of Delegation	John Hill	hill@prc.unisys.com
Canadian HoD	Arnie Powell	arniep@canvm2.vnet.ibm.com
UK HoD	David Cannon	cannon@exeter.ac.uk
German HoD	Ron Elliot	elliott%aixsm@uunet.uu.net
Dutch HoD	Herman Wegenaar	(phone: +31 50 637052)
Japanese HoD	Nobuo Saito	ns@slab.sfc.keio.ac.jp
French HoD	Jean-Michel Cornu	jean-michel.cornu@afuu.fr
Danish HoD	Keld Simenson	keld@dkuug.dk
New Zealand HoD	Keith Hopper	kh@waikato.ac.nz

Report on POSIX.0: The POSIX Guide

Kevin Lewis <klewis@gucci.ent.dec.com> reports on the January 11-15, 1993 meeting in New Orleans, LA:

First off, let me say that this will be a relatively short report given that the group's exclusive activity currently is ballot resolution. To fulfill my promise made from last meeting, I have provided a more detailed balloting profile. It is as follows:

Vote	Ballots
affirmative	28
negative	30
abstentions	11

Response	Count
objections	494
comments	640
total	1134

There were 69 ballots returned (85%) of which 48% were affirmative.

The January meeting was a rolled-up sleeves session where the group focused totally on ballot resolution. It was a bit of a struggle because we transitioned some section leaders. New people took over for some, and others were absent.

It became apparent by week's end that we wouldn't resolve all the ballots. It also became apparent that, as this process continues, a more authoritarian role on the part of the section leaders and the chair will be necessary to expedite ballot resolution.

The ballot-resolution group agreed that section leaders should use electronic means to survey the group on issues where they feel such help is needed.

The next deadline will be March 8, at which time all ballot resolutions will be submitted to the ballot coordinator. The ballot coordinator will work with the technical editor to prepare an interim draft for the April meeting as a last look by the whole group before it goes out for recirculation.

The group attempted (as it is prone to do at times) to step back onto some old ground, re-opening discussions that had been resolved or falling down potential rat holes. On these occasions, the chair had to bring the the group back in to the process of resolving the ballot at hand. There is a balancing act that the group must maintain. On one hand, there is the desire to ensure that the document does not go out with any major defects. On the other hand, we need to keep the resolution process moving forward. This sometimes compels the group to open up broader discussions than are really necessary.

The group consensus is to err on the side expediting the process in order to get this work into the hands of the balloting group, which has been asking for it.

Report on POSIX.2: Shell and Utilities

David Rowley <david@mks.com> reports on the January 11-15 meeting in New Orleans, LA:

A Brief Update

The POSIX.2 standard (IEEE Std 1003.2-1992) should be available from the IEEE in April as a 2-volume set for $95. The standard consists of both the "Dot 2 Classic" and "Dot 2a" components, previously balloted as separate standards. The IEEE Standard (based on Draft 12 from the ballot group) is identical (at least from a technical standpoint) to ISO/IEC Draft International Standard 9945-2:1992.

NIST expects to issue the new draft FIPS (Federal Information Processing Standard) for POSIX.2 early in April, with the final version expected by late 1993.

POSIX.2b work continues, now on draft 5. The group is still wrestling with the ISO 1001 tape format for PAX .

Test method development for the base POSIX.2 standard nears completion, and a full recirculation of the P2003.2 document is expected by early summer.

X/Open has awarded the role of integrator for the combined POSIX.2 / XPG4 Commands and Utilities test suite project to a joint venture between BSI (British Standards Institute) and Mindcraft, Inc. (Palo Alto, CA). The suite is expected to be available early in 1994.

Background

A brief POSIX.2 project description:

- The base utilities of the POSIX.2 standard deal with the basic shell programming language and a set of utilities required for the portability of shell scripts. It excludes most features that might be considered interactive. POSIX.2 also standardizes command-line and function interfaces related to certain POSIX.2 utilities (e.g., *popen()*, regular expressions, etc.). This part of POSIX.2 , which was developed first, is sometimes known as "Dot 2 Classic."

- The User Portability Utilities Option, or UPUO, is an option in the base standard (previously known as POSIX.2a). It standardizes com-

mands, such as *vi*, that might not appear in shell scripts, but are important enough that users must learn them on any real system.

Some utilities have both interactive and non-interactive features. In such cases, the UPUO defines extensions from the base POSIX.2 utility. Features used both interactively and in scripts tend to be defined in the base utility.

- POSIX.2b is a project that covers extensions and new requests from other groups, such as a new file format for PAX and extensions for symbolic links. It also includes resolution of items arising from comments by ISO Working Group 15.

POSIX.2 is equivalent to the International Standards Organization's ISO DIS 9945-2 – the second volume of the proposed ISO three-volume POSIX standard.

Report on POSIX.5: Ada Bindings to POSIX.1

Del Swanson <dswanson@mhs.sp.paramax.com> reports on the January 11-15, 1993 meeting in New Orleans, LA:

The POSIX.5 working group has been working to produce Ada language bindings to POSIX standards. The Ada binding for POSIX.1, IEEE Std 1003.5-1992 (aka POSIX.5), has now been published as an IEEE standard. Suitable kudos were spread around to the contributors at the January meeting in New Orleans. The next target is the development of bindings to the Real-Time Extensions standards being developed by the POSIX.4 group.

The binding to POSIX.4 is relatively straightforward. A draft thin binding to POSIX.4 was prepared by one of our members on contract to the U.S. government. This draft has now been updated by the group, and massaged into IEEE format. This document, POSIX.20 (draft 1), was circulated for mock ballot in December, with comments due in by February 4. Our goal is to go to real ballot soon after the April meetings, and have POSIX.20 approved as a standard hard on the heels of POSIX.4 LIS (Programming-Language-Independent Specification).

Meanwhile, work has begun concurrently on the binding to POSIX.4a (threads extensions). An initial draft has been prepared, and was debated at the January meeting. Significant changes to it are now expected to be put on hold until the next version of POSIX.4a appears. The POSIX.5 group met with the POSIX.4 group in January to get an update on the status of the threads work.

Orthogonal to this update, some members of the POSIX.5 group are becoming concerned about the relationship of the threads interface and the updates to the Ada language standard that is commonly called Ada 9x. Some significant changes and enhancements are expected in the tasking model for Ada 9x, and in some respects, they have an adverse impact on the ability to implement an Ada runtime library using POSIX threads. These concerns are being provided to the POSIX.4 group, for consideration in the ballot resolution process.

In January, we also met with a delegation of the group that is formulating the POSIX.1 LIS. Several members of the POSIX.5 group had objected to a few points in the ballot. In the discussion, general agreement was reached on issues of naming, I/O formulations, and implications of concurrency within POSIX processes. Signal concerns were also discussed, but it remains to be seen whether mutually agreeable formulations result.

The core of the issue is that Ada runtime libraries require the exclusive use of a few of the signals to implement Ada scheduling and exception delivery. The Ada binding to POSIX.1 specifies this, and it is our perspective that the LIS should allow such exclusivity.

Some members of the group have been facing problems with the IEEE standards office related to the copyright of the Ada binding. We have also been receiving reports from several others of similar problems. The copyright, of course, states that none of the material in the standard may be reproduced in any fashion without the permission of the IEEE.

Well, how does one compile an Ada package specification (or a C header file, for that matter), which happens to be part of the standard, without copying it, and reproducing it in the file system of a computer? How does one introduce the implementation-defined values in appropriate places? How does one inform one's users about the values so defined? How does one provide access to these specifications, so that calls to the interfaces can be compiled in application code?

At first, a couple of people applied for, and received, official limited permission at least to make compilable copies for development purposes. Our group was concerned about the reticence, the bother of individual application, and the implications for the ease of use of the standard. Therefore our chair approached the IEEE, explained the situation, and appeared to have reached an amicable agreement.

The IEEE defined a policy of approval to copy the specifications for such use, with requests and automatic approval exchanged by email. Copies

of the correspondence were to be sent to a member of POSIX.5 to monitor the process. To date, upwards of 20 requests have been monitored, but no approval responses have been forthcoming. We have not heard of similar difficulties for other language bindings. Obviously, this is a serious problem, and will be addressed further at the next meeting.

On a more technical level, the April meeting will be spent resolving comments to the mock ballot; defining strategy on the coordination of the POSIX.5 standard with the revised POSIX.1 and the proposed POSIX.20 (Ada binding to the real time extensions), and addressing the issues of interpretations that have been raised with POSIX.5.

Report on POSIX.7: System Administration

Bob Robillard <duke@cc.bellcore.com> reports on the January 11-15, 1993 meeting in New Orleans, LA:

Introduction

Three of the POSIX.7.1 System Administration small groups met during the week:

POSIX.7.1 – Printing Administration
POSIX.7.2 – Software Installation and Management, and
POSIX.7.3 – User and Group Administration.
There were also several plenary meetings of the group, and issues were discussed that cut across subgroups.

POSIX.7 – Overall

The first issue discussed by POSIX.7 as a whole was the question of Test Assertions (TA) and Language Independence (LIS). I suspect this issue is discussed at length in another snitch report, so I'll just give POSIX.7's angle. The group had discussed this in the past, and was clearly in agreement with Stephen Walli's movement to drop these requirements. We wrote a letter to the SEC stating our position and POSIX.6 (Security Extensions), POSIX.11 (TP Profile), and POSIX.14 (Multiprocessor Profile) co-signed it.

Since the Test Assertion requirement was suspended by the SEC and the Language Independence requirement is under attack, the group has decided to limit the amount of time spent on these to a bare minimum. If an LIS is still required by the time the Print Group goes to ballot in May, a rough draft of one will be submitted with the real, C language draft.

The second cross-group issue debated was the question of using common command line options for "extended options " (i.e., options that take more than a simple switch to specify). Both the Printing and Software Management command line interfaces (CLI) describe similar files that can contain extended options for commands. It was decided to use the same option letter for these "extended options files" (-X).

Both CLIs also allow these extended options to be passed in a quoted string on the command line, and there was an agreement to use the same letter for this as well (-x). Unfortunately, the groups couldn't agree upon a common syntax for the content of the files and strings.

The last POSIX.7-wide issue was the question of distinguished names. These are names of entities in the network, e.g., machines or print daemons. It was decided that the POSIX.7.X drafts would require that implementations accept Internet style names and can allow any other style they want. The suggestion of requiring X.500 style names (`/co=usa^/org=-dec^/host=foobar^/printer_server1`) was voted down, mainly because it's not widely used. In fact, even the POSIX X.500 API doesn't use it directly! That API requires applications to parse the name given on the line and fill a C structure, so it is just as happy with Internet names as with the "/" style names.

POSIX.7.1 — Print Administration

The first issue the Printing Group dealt with was the forthcoming new edition of the ISO Document Printing Application (DPA) Draft. The POSIX printing document is based on the ISO DPA. A new version of the ISO DPA is due out in May or June, and the printing group had to decide how to deal with the new document.

One of the members of POSIX.7.1 is also a member of the ISO DPA committee. He went over the changes in the next version, and the group decided that their effects on the POSIX document were small enough that they could be incorporated into the draft during the ballot process.

The group next discussed a number of changes proposed by the Open Software Foundation. None of these were serious changes, and most were adopted. In addition, the section describing the Name Service was extensively rewritten and the programming examples in the rationale for the API were brought up to date.

The printing group started the process of forming a ballot group. Although the dates for ballot are not final, the POSIX.7.1 ballot will probably run from May 10 to July 16. To get on the ballot group, contact:

IEEE Standards Office
Attn: Anna Kaczmarek
PO Box 1331
Piscataway, NJ 08855-1331 USA

Tell her you want to join the "TCOS Standards Subcommittee." [TCOS, *the Technical Committee on Operating Systems, is no longer the parent of the POSIX standards subcommittee.* TCOS-SS *has become PASC, the Portable Applications Standards Committee. – Ed.*] Give your IEEE or IEEE Computer Society Number, if you've got one. Only IEEE or Computer Society members are eligible balloters on IEEE proposed standards; nonmembers can participate as Parties of Interest, which means they can vote and object, but their vote doesn't count in the final tally.

Alternatively, you can contact Bob Robillard (908/644-2249, *<duke@cc.bellcore.com>*).

POSIX.7.2 — Software Management

The Software Group had a set of written comments from a number of the members, and they spent the week going through them, improving the draft for their mock ballot. The mock ballot will be conducted from March 1 to March 31. To join in this ballot, contact Jay Ashford, *<ashford@austin.ibm.com>*, 512/838-3402.

Many of the comments reviewed dealt with cleaning up the command line interface (e.g., determining options names, and so on). There was a long debate on the value of allowing multiple "MIBs" or databases of installed software packages. In the end, the group decided to permit this.

Other details were worked out, such as the use of a name server, the media format (the POSIX *pax* format was chosen), and use of environment variables. The idea of making everything in the standard optional except for the distribution format was discussed. This would ensure portability of distributions, but wouldn't do anything toward promoting a common command set. The decision was reached to make the entire draft required, at least for the present.

POSIX.7.3 — User and Group Management

The User/Group subgroup made some concrete progress toward a real draft. After reviewing POSIX.1 and POSIX.2 for any user/group items and meeting with POSIX.6 to learn their concerns, they wrote a scope and picked three base documents to merge into a draft:

- USL's System V Interface Definition, 3rd Edition (with additions from the new Distributed Manager user management product)
- OSF's DCE user management
- SCO's user management product

The Scope and Base document list was given to POSIX.7's PMC Mentor, who agreed that they would make a good start. *[The POSIX Project Management Subcommittee (PMC) assigns someone to act as a mentor or guide to each project, who is supposed to be a shield for some of the procedural work, and help the project keep on track.– Ed.]*

POSIX.7.3 will be creating a Project Authorization Request (i.e., permission to start a document) in April. The PMC Mentor is happy with their proposal, and intends to recommend granting approval. In anticipation of that, they will attempt to have Draft 1 of POSIX.7.3 in the mailing before the April meeting.

While it is somewhat premature to work out the details of the command line, POSIX.7.3 contributed to the joint debate on the "extended options" (i.e., -x and -X) and intends to follow the lead of the other two groups.

In addition, they presented the idea of another common option: a "template" object, used as a template from which to create real objects. For example, a typical-user template would have all the information necessary to set up a new typical user, and could be specified in the useradd command (this is similar to inheritance in the object oriented world). There was agreement from POSIX.7.1 and POSIX.7.2 that this could be useful, and will be investigated further.

Report on POSIX.18: POSIX Platform Environment Profile

Paul Borman <prb@cray.com> reports on the January 11-15, 1993 meeting in New Orleans, LA:

The title of POSIX.18 reads: *"Draft Standard for Information Technology – POSIX Standardized Profile – USI-P001 POSIX Platform."*

The title says it all. What is a POSIX Standardized Profile? What does USI-P001 mean? If you can answer these questions – *We Want You!* Unfortunately, this confusion is carried through the whole draft.

Due to problems of redundancy and obfuscation, the working group unmercifully hacked away at the draft with an axe in the previous meeting. This meeting we took out our Bowie knives to whittle it down further still.

The three major issues discussed were the prose, what was it for, and what new normative text or changes to the normative text should be made.

This discussion of the prose centered around the large amounts of redundant and apparently meaningless text in the draft. Was it boiler plate?

Was it just the previous editor? Did it simply come from Planet X in the middle of the night? Extraterrestrial or not, either the prose was simply removed or we reworded it to be more easily understood.

First on the list of things to be clarified was the introduction, which was determined to be mostly redundant or irrelevant. We did decide to reword it to indicate that POSIX.18 describes UNIX Classic or Version 7, for those who remember it. The profile still will not define administrative interfaces, or even a way to login.

We did lobby the POSIX.1a working group to modify a couple of interfaces to bring them in line with FIPS 151-2. [*FIPS 151-2 is the updated NIST specification of the POSIX.1 standard, used in U.S. government procurements where POSIX-like functionality is required.– Ed.*] We hope POSIX.18 will mirror this new FIPS. These modifications were:

- When *read()* or *write()* is interrupted by a signal, after having read/written any data, then they will return the byte count instead of -1.
- That the group-ID of a file at creation time is that of the directory in which it is created, and not the effective group-ID of the process.

We introduced text in POSIX.18 that requires that CS7, CS8, CSTOPB, PARODD, and PARENB be supported from the POSIX.1 General Terminal Interfaces section.

We are not clear exactly what NIST was trying to accomplish by this and any comments would be appreciated.

There were several parts of the document that existed to fulfill TR10000 requirements, but as TR10000 is changing, much of this was removed. [*TR10000 is the ISO technical report, defined originally in the OSI profile world, and now making itself felt in the POSIX profile space. – Ed.*] We are going to lobby for the new TR10000 to require less and make it easier to understand.

We restructured the two or three pages of real normative text in the document in line with our decision in the last meeting to require the C language.

Due to a new SEC ruling, we plan to remove the current, inadequate test assertions in the document, and concentrate on the normative text.

Our major additions to the normative text, aside from the FIPS 151-2 item mentioned earlier, were coherency statements. We have required, for example, that all the base standards that are pointed to by this profile must be implemented with the the same file-system name space and use a consistent byte size.

We also mandated that text files would be usable between all the different base standards and that text files can be used to contain source code that the compilers can compile. Without these sorts of statements it would have been technically possible to have a conforming system in which *vi* was not capable of creating a file that the C compiler could compile!

Other things were that the shell could execute a program built with the compilers and that the compiler would allow use of the POSIX.1 functions. Pretty straightforward and obvious stuff, but that is the sort of thing a profile must point out to make itself useful.

Overall, I feel that the POSIX.18 draft made a lot of forward progress, but because it now references POSIX.1a it cannot go to ballot. We also feel we need to do a bit more work cleaning up the wording of the draft (and come to grips with what NIST is really asking in FIPS 151-2).

Please note that POSIX.18 is the profile that will more than likely define the basics of a time-sharing UN*X system in the future. If you are concerned about this, you might want to show up at our next meeting, and you will certainly want to join the balloting group.

Report on ANSI X3B11.1: WORM File Systems

Andrew Hume <andrew@research.att.com> reports on the Dec. 14-16, 1992 meeting in Orlando, FL

Introduction

X3B11.1 is working on a standard for file interchange on random-access optical media: a portable file system for WORMs or rewritable optical disks. TC15 is a committee within ECMA that works on file system standards. This report covers the last two X3B11.1 meetings. In brief, our ECMA standard has been published, we have entered the fast-track process, and are now DIS 13346!

ECMA -167

I won't describe ECMA-167 again; if you want the gory details, see my last snitch reports. At the time of my last report, the ECMA General Assembly had approved ECMA-167 as a standard and "all" we had to do was publish it. This was not an entirely smooth process, but it could have been worse.

The source of the draft was a weird form of text that, after processing by several *awk* and *sed* scripts, became more or less normal *troff -ms* input. The ECMA office uses a popular PC pub-

lishing package. The conversion was mostly done mechanically (using RTF as the intermediate form) with our chair Ed Beshore doing the final pass by hand on his PC before sending floppies off to Geneva. A mere three galley proofs later, I (as technical editor) approved the current draft. Proofing galleys is about as tedious as it sounds. (It's good to do while watching Sunday afternoon football.) I was ably assisted by Howard Kaikow, now no longer at DEC. The draft was much improved stylistically by this process, although I personally find the ECMA house format to be visually unappealing.

International Activity

There is substantial international interest in volume and file structure standards, particularly for removable optical media. That is why our committee has an ISO standard as its main goal, rather than an ANSI standard. That is also why we have bent over backwards to solicit input from, and work with, Europe (ECMA), Japan (JNC), and ISO (SC15).

We were very pleased to learn that ECMA-167 is now DIS 13346. The six-month ballot period will end July 28, 1993 and the special working group meeting that addresses the ballot responses has been tentatively scheduled for October 13-15, 1993 in Geneva, Switzerland. The end is definitely in sight.

The other activity going on in SC15 is work on a reference model for information interchange between open systems by interchangeable storage media. This is similar to the OSI reference model; in fact, rather too similar in my mind. Although reference models can be astonishingly boring, a good one would have helped the development of our standard a little, and a bad one can easily hinder the development of good standards. The current draft of the reference model represents early work and is being commented on by interested parties in our committee and by an ad hoc group in X3B8.

Future Activity

The committee's focus is now split among three areas. The first area is preparing for voting on DIS 13346. This is fairly routine but intricate because of procedural rules and delays within the U.S.; documents have to get passed from ISO to ANSI to X3 to X3B11 and finally to us. We vote on a recommendation for the U.S.'s vote, and then that goes back up the chain. The complications involve meeting schedules, voting deadlines and making sure no one inadvertently says no.

The second area is implementing ECMA-167. I know of five implementation efforts; one commercial implementation is beta testing with customers. As a means of verifying our understand-ing of the standard, and as a way of improving the level of interchange, Hewlett-Packard organized a meeting on conformance testing for ECMA-167 in February in Fort Collins, CO. This was surprisingly popular, with about 30 companies attending. In brief, the meeting agreed to work on the areas of conformance testing, and the details of how to translate between conforming media and various operating systems' interfaces.

The third area is addressing work for future standardization. This includes specific proposals for issues like compression, which ECMA-167 supports in a generic way, and proposals for niche targets with specific reliability and performance goals. This work is parallel to, and asynchronous with, the progress of DIS 13346. If anyone has specific proposals for things not adequately addressed in ECMA -167, they are invited to make them known to X3B11.1. (If you can't or don't want to attend meetings, I may be willing to be an advocate for you!) Contact Ed Beshore for meeting details.

Electronic Distribution of Standards/Drafts Several

X3B11.1 documents have been available electronically by both *ftp* and email (*netlib*) from <*research.att.com*>. (For *ftp*, login as *netlib*.) For details, get *index* from *research/memo*. The main documents are:

- The standard itself (121 pages including TOC and index). (This is the actual standard as published; ECMA has approved its electronic distribution.)
- A technical overview (12 pages). This gives a high-level overview, but has significant technical content.
- A programmer's guide (20 pages). A low-level guide through the standard from a C programmer's point of view. It gives you enough details to design an implementation and do most of the implementation.

Finale

If you would like more details on X3B11.1's work, you should contact either me at <*andrew@research.att.com*>, 908/582-6262) or the committee chair, Ed Beshore at <*edb@hpgrla.gr.hp.com*>, 303/350-4826). The next two meetings are in Tucson in mid-March and Long Island in mid-July. Anyone interested in attending should contact Ed Beshore.

User Support Mailbox

Dear Unixsupport,

Is there any easy way to insert a carriage return after every, say, 10th line of a file?

Crowded

Dear Crowded,

You could certainly do it in `awk`, and probably in `sed`. But here's a `perl` of a solution.

```
#!/usr/bin/perl

$USAGE = "USAGE: cr-after-n posint [filename...]";

$n = shift @ARGV || die $USAGE;
$n > 0 || die $USAGE;

$count = 0;
while (<>)
{
   print;
   if (++$count == $n)
   {
      print "\n";
      $count = 0;
   }
}
exit 0;
```

Dear Unixsupport,

This Solaris is weird. Well, different to SunOS 4, anyway. Is there any way my shell can tell whether the machine is running it, so I can, for example, put `/usr/ccs/bin` in my search path?

Sunburnt

Dear Sunburnt,

The `uname` command can help you tell SunOS versions apart. The Solaris "operating environment" (yes, that's what they call it) uses SunOS 5. I use the following in my `.cshrc` file. (If you're not using `csh`, write the equivalent in your own shell.)

```
# Solaris?
set SOL = 0
if ('uname -r | cut -d. -f1' == '5') set SOL = 1
```

Then you can say things like

```
if ( $SOL ) then
   setenv PATH /usr/bin:/usr/ucb:/usr/ccs/bin
else
   setenv PATH /usr/ucb:/usr/bin
endif
```

and

```
if ( ! $SOL ) alias psu 'ps axu | head -15'
```

Janet Jackson <janet@cs.uwa.edu.au>
From WAUG, the WA Chapter of AUUG

AUUG MANAGEMENT COMMITTEE

SUMMARY OF MINUTES OF MEETING 23rd April 1993

Present: Frank Crawford, Glenn Huxtable, Rolf Jester, Phil McCrea, John O'Brien, Michael Paddon, Greg Rose, Peter Wishart, and Liz Fraumann.

Guests: Wael Foda, Lachie Hill, Piers Lauder, Alan Taylor

Apologies from Chris Maltby

1. AUUG '93

Piers Lauder (PL), Programme Chair AUUG '93 reviewed the status of the conference. Over 50 papers were submitted. It would yield approximately a 2:1 ratio of reject vs accept. It will ensure a higher standard of quality in the papers.

FC suggested that rejected papers be submitted for publication in AUUGN.

2. REPORTS

2.1. Membership Report

Over 100 non-financial members were dropped from the membership list. This is a significant number and caused some concern for the committee.

PM suggested a Membership Drive. The following "brainstorm" ideas followed:

(1) Must offer benefits for members

(2) The local chapter activities

(3) Chapters should advertise their activities in advance

(4) There is a need for a chapter in NSW

(5) Should cater to the unemployed, ie like students

(6) Target the top

(7) PR campaign - create awareness

Institutional member benefits were reviewed and possible new benefits were suggested.

2.2. AFUU CONFERENCE REPORT

The AFUU (French Unix Users Group) conference went very well. We still need confirmation/information on the speakers from AFUU who will participate in AUUG93.

2.3. PRESIDENT'S REPORT

The Fiji computer conference went well. Several individuals enquired about AUUG. PM suggested we extend the AUUG member discount to Fiji Computer Society members.

GGOS (Government Guide to Open Systems) - has been delayed. Modifications were being made to the document.

ACS is also sponsoring an ASWEG conference at the Hilton in Sydney the same week as AUUG. There are many small conferences which ACS sponsors via name only and this is just an example of this

situation. This specific event has been on going for the past several years.

2.4. SECRETARY'S REPORT

The Adelaide summer conference was well organised and well attended, 40 - 50 attendees. They were "busting out of the venue," and will likely need to locate a different venue next year. They had interesting speakers and good local participation. It showed a strong community involvement. They have scheduled 3-4 general meetings for this year.

AUUG wrote to the SUG group re: the AARNet connection. They have been given a free subscription for a 1 year period, and then should look to future activities through participation via a chapter. Response time out is 1 July 1993.

2.5. TREASURER'S REPORT

FC reported the cash management account is currently holding $80K and the checking account is holding $16K. It was noted the fiscal year ends 31 May. FC presented the budget as it stands for the years' end. FC presented the draft budget for 93/94.

3. BUSINESS CARRIED OVER

PW reviewed the insurance quotation for AUUG Inc. and its activities. For $2-5 M in coverage the price is $375 - $440 per year. $100 excess on claims. PW will secure additional quotations and proceed based on $5M worth of coverage.

4. OTHER BUSINESS

4.1. Nominations for Elections

With the current nominations we are 2-3 committee members short. However due to the competition for postitions, an election must be held.

Nominee	Position(s)
Michael Paddon -	President, VP, Secretary, Treasurer, Committee
Greg Birnie -	Committee
Glen Huxtable -	VP, Committee
Chris Maltby -	Committee
Phil McCrea -	President
Peter Wishart -	Secretary, Committee
Frank Crawford -	Treasurer, Committee, Returning Officer, Asst. Returning Officer

Ballots will be issued the first week of May.

4.2. LOCAL CHAPTER PETITIONS

The following chapters presented formal petitions to be accepted as an AUUG chapter:

(1) SA - Approved.

(2) NT - subject to receipt of actual petition with signatures. Approved.

(3) QLD - Approved.

(4) WA - Approved (see discussion below).

WAUG (the West Australian UNIX Group) is an exitsing UNIX group in WA which is changing into the WA Chapter of AUUG. WAUG has existing members and the transfer of WAUG's commitement to those members from WAUG to the WA chapter of AUUG presented some special issues which were discussed by the committee.

It was moved and passed that WAUG members will have the option to pay the difference ("top-up" fee) of $38 for the individual membership. Institutional WAUG members rate would pay the difference ~$200, or the $38x2 to become (2) individual members. In keeping with the current AUUG membership terms, which have been aligned to expire in June and December of each year, WAUG memberships which are topped up to full AUUG membership will also be extended to the following June or December. It was noted that those WAUG members not choosing to take the option could still receive those chapter benefits which are provided directly by the WA Chapter of AUUG.

It was felt that outlying (i.e. those not in Sydney/Melbourne) chapters needed to receive additional funding. It was moved and passed that "Our policy of funding local chapters is that 50% of the AUUG membership fee will be returned directly to the chapters in : WA, NT, QLD, and SA."

4.3. LOCAL CHAPTER COUNCIL

The first council meeting will be held the day before or after the executive committee in Sydney meeting in August. The committee are expected to attend this first meeting.

5. NEXT MEETING

2nd July 1993, 10:00am, Softway. This will be a combined meeting of the new and old committee since it is the first after the election for 1993/94.

-- Peter

AUUG Membership Categories

Once again a reminder for all "members" of AUUG to check that you are, in fact, a member, and that you still will be for the next two months.

There are 4 membership types, plus a newsletter subscription, any of which might be just right for you.

The membership categories are:

Institutional Member
Ordinary Member
Student Member
Honorary Life Member

Institutional memberships are primarily intended for university departments, companies, etc. This is a voting membership (one vote), which receives two copies of the newsletter. Institutional members can also delegate 2 representatives to attend AUUG meetings at members rates. AUUG is also keeping track of the licence status of institutional members. If, at some future date, we are able to offer a software tape distribution service, this would be available only to institutional members, whose relevant licences can be verified.

If your institution is not an institutional member, isn't it about time it became one?

Ordinary memberships are for individuals. This is also a voting membership (one vote), which receives a single copy of the newsletter. A primary difference from Institutional Membership is that the benefits of Ordinary Membership apply to the named member only. That is, only the member can obtain discounts an attendance at AUUG meetings, etc. Sending a representative isn't permitted.

Are you an AUUG member?

Student Memberships are for full time students at recognised academic institutions. This is a non voting membership which receives a single copy of the newsletter. Otherwise the benefits are as for Ordinary Members.

Honorary Life Membership is not a membership you can apply for, you must be elected to it. What's more, you must have been a member for at least 5 years before being elected.

It's also possible to subscribe to the newsletter without being an AUUG member. This saves you nothing financially, that is, the subscription price is greater than the membership dues. However, it might be appropriate for libraries, etc, which simply want copies of AUUGN to help fill their shelves, and have no actual interest in the contents, or the association.

Subscriptions are also available to members who have a need for more copies of AUUGN than their membership provides.

To find out your membership type, examine your membership card or the mailing label of this AUUGN. Both of these contain information about your current membership status. The first letter is your membership type code, M for regular members, S for students, and I for institutions, or R for newsletter subscription. Membership falls due in January or July, as appropriate. You will be invoiced prior to the expiry of your membership.

Check that your membership isn't about to expire and always keep your address up-to-date. Ask your colleagues if they received this issue of AUUGN, tell them that if not, it probably means that their membership has lapsed, or perhaps, they were never a member at all! Feel free to copy the membership forms, give one to everyone that you know.

If you want to join AUUG, or renew your membership, you will find forms in this issue of AUUGN. Send the appropriate form (with remittance) to the address indicated on it, and your membership will (re-)commence.

As a service to members, AUUG has arranged to accept payments via credit card. You can use your Bankcard (within Australia only), or your Visa or Mastercard by simply completing the authorisation on the application form.

MEMBERSHIP APPLICATION
INDIVIDUAL

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 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, and remember to select either surface or air mail.

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 surface mail	☐ $ 20.00
International air mail	☐ $ 60.00
UniForum affiliate membership	☐ $ 20.00

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

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

_____
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

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 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, and remember to select either surface or air mail.

We, _____
(Company Name)

do hereby apply for:

Renewal/New* Inst. membership of AUUG	☐ $350.00
International surface mail	☐ $ 40.00
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.

AUUG Incorporated
Application for Newsletter Subscription
AUUG Inc.

Non members who wish to apply for a subscription to the Australian UNIX systems User Group Newsletter, or members who desire additional subscriptions, should complete this form and return it to:

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

- Please don't 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, or credit card authorisation, and remember to select either surface or air mail.
- Use multiple copies of this form if copies of AUUGN are to be dispatched to differing addresses.

This form is valid only until 31st May, 1994

Please *enter / renew* my subscription for the Australian UNIX systems User Group Newsletter, as follows:

Name: ..... Phone: ..... (bh)

Address: .... (ah)

..... Net Address: .....

.....

..... *Write "Unchanged" if address has*

..... *not altered and this is a renewal.*

For each copy requested, I enclose:

☐ Subscription to AUUGN	$ 90.00
☐ International Surface Mail	$ 20.00
☐ International Air Mail	$ 60.00

Copies requested (to above address) _____

Total remitted AUD$_____

(cheque, money order, credit card)

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

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

Account number: _____ _____ _____ _____. Expiry date: __/__.

Name on card: _____ Signed: _____

Office use only:

Chq: bank _____ *bsb* _____-_____ *a/c* _____ *#* _____

Date: __/__/__ *$* *CC type* ___ *V#* _____

Who: _____ *Subscr#* _____

AUUG

Notification of Change of Address

AUUG Inc.

If you have changed your mailing address, please complete this form, and return it to:

AUUG Membership Secretary
P O Box 366
Kensington NSW 2033
Australia

Fax: (02) 332 4066

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

Old address (or attach a mailing label)

Name: ..... Phone: ..... (bh)

Address: .... (ah)

..... Net Address: .....

.....

.....

.....

New address (leave unaltered details blank)

Name: ..... Phone: ..... (bh)

Address: .... (ah)

..... Net Address: .....

.....

.....

.....

Office use only:

Date: ___/___/___

Who: _____ *Memb#* _____

The Australian Financial Review's Open Systems survey on Monday, June 28 demonstrates how Australia has embraced the freedoms offered by this new technology without much of the cynicism of some in the computer industry.

- The Australian Financial Review will look at how the Federal Government has seen the benefits in a computing community not tied to one particular supplier and has committed to a policy of implementing open systems.
- The survey will investigate the reality behind the marketing hype of open systems, as well as the computer companies who are offering their systems within an open systems framework.
- For an account of the successful users of open systems and their suppliers, and a look into the future, turn to the Open Systems survey in The Australian Financial Review on June 28.

SURVEY RATES		MECHANICAL REQUIREMENTS
		Column Measurements
Full Page $9,480.00	22 x 4 ..... $3,476.00	1 column ..... 4.1 cm
5 columns $7,900.00	22 x 3 ..... $2,607.00	2 columns ..... 8.7 cm
3 columns $4,740.00	20 x 3 ..... $2,370.00	3 columns ..... 13.2 cm
2 columns $3,160.00	20 x 2 ..... $1,580.00	4 columns ..... 17.7 cm
30 x 5 ..... $5,925.00	18 x 3 ..... $2,133.00	5 columns ..... 22.3 cm
30 x 4 ..... $4,740.00	18 x 2 ..... $1,422.00	6 columns ..... 26.9 cm
30 x 3 ..... $3,555.00	16 x 3 ..... $1,896.00	Depth of column: 39.0 cm
30 x 2 ..... $2,370.00	16 x 2 ..... $1,264.00	ADVERTISING PRODUCTION CHARGE
28 x 3 ..... $3,318.00	14 x 2 ..... $1,106.00	$6.50 per single column centimetre
26 x 4 ..... $4,108.00	12 x 2 ..... $948.00	BOOKING AND CANCELLATION DEADLINE
26 x 3 ..... $3,081.00	10 x 2 ..... $790.00	Thursday, June 17
24 x 3 ..... $2,844.00	8 x 2 ..... $632.00	MATERIAL DEADLINE
		Thursday, June 24

THE AUSTRALIAN
FINANCIAL REVIEW

Fairfax. Well written. Well read.

*Source: Roy Morgan September 1992