



Australian UNIX systems User Group Newsletter

AUUGN

Volume 12, Number 6

December 1991

AMERICAN

Volume 12 Number 6

December 1991

The AUUG Incorporated Newsletter

Volume 12 Number 6

December 1991

CONTENTS

AUUG General Information	3
Editorial	5
AUUG Institutional Members	7
President's Report	9
AUUG 1992 Summer Conference Series	11
Open System Publications	12
SESSPOOLE	13
ACSnet Survey	14
AUUG Book Reviews	17
AUUG Book Club Reviews	20
AUUG Book Club Order Form	25
Multiprocessor Streams for Plan 9	26
A Scientific Visualization Tool	36
Jeeves, the Butler	40
From ;login - Volume 16, Number 4	51
An Update on UNIX-Related Standards Activities	51
Management Committee Minutes - 9th DECEMBER 1991	72
AUUG Membership Categories	76
AUUG Forms	77

Copyright © 1991 AUUG Incorporated. All rights reserved.

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.

Copying without fee is permitted provided that copies are made without modification, and are not made or distributed for commercial advantage. Credit to AUUGN and the author must be given. Abstracting with credit is permitted. No other reproduction is permitted without prior permission of AUUG Incorporated.

* 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.

All correspondence concerning membership of the AUUG should be addressed to:-

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

Phone: (02) 361 5994
Fax: (02) 332 4066

General Correspondence

All other 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 Executive

President	Pat Duffy <i>pzd30@juts.ccc.amdahl.com</i> Amdahl Pacific Services Pty. Ltd. 1 Pacific Highway North Sydney NSW 2000	Vice-President	Chris Maltby <i>chris@softway.sw.oz.au</i> Softway Pty. Ltd. 79 Myrtle Street Chippendale NSW 2008
Secretary	Rolf Jester <i>rolf.jester@sno.mts.dec.com</i> Digital Equipment Corporation (Australia) Pty. Ltd. P.O. Box 384 Concord West NSW 2138	Treasurer	Frank Crawford <i>frank@atom.ansto.gov.au</i> Australian Supercomputing Technology Private Mail Bag 1 Menai NSW 2234
Committee Members	Andrew Gollan <i>adjg@softway.sw.oz.au</i> Softway Pty. Ltd. 79 Myrtle Street Chippendale NSW 2008		Glenn Huxtable <i>glenn@cs.uwa.oz.au</i> University of Western Australia Computer Science Department Nedlands WA 6009
	Peter Karr Computer Magazine Publications 1/421 Cleveland Street Redfern NSW 2016		Michael Tuke <i>mjt@anl.oz.au</i> ANL Ltd. 432 St. Kilda Road Melbourne VIC 3004
	Scott Merrilees <i>Sm@bhpes.oz.au</i> BHP Information Technology P.O. Box 216 Hamilton NSW 2303		

AUUG General Information

Next AUUG Meeting

The AUUG 1992 Summer Conference Series are to be held between February and April 1992 (see later in this issue for more details).

The AUUG'92 Conference and Exhibition will be held from the 8th to the 11th of September, 1992, at the World Congress Centre, Melbourne.

AUUG Newsletter

Editorial

Finally the last issue of AUUGN for 1991 is out. Once again sorry for the delay, but as the previous editors experienced, I find that I get a number of promises for articles but getting these articles is a slow process and deadlines seem not to be observed. As I have mentioned in previous issues I need papers from people working in the area of Open Systems. I am always willing to help get it in a suitable format for printing.

In this issue I have another article on Plan 9 to follow those from AUUG'91 and previous AUUGNs. I also have an article from Jack Dikian who tells us a bit about an important product he is working with and one paper from Bernd Felsche which was presented at the Summer technical meeting last year. Hopefully we will have lots more from this years Summer meeting in future issues.

Also in this issue we have restarted the book reviews. Thank you to the people that reviewed them. Now that book reviews have been started again, anyone interested in reviewing books should contact Dave Newton. As some people seem not to be aware, the book that you review is yours to keep.

Jagoda Crawford

AUUGN Correspondence

All correspondence regarding the AUUGN should be addressed to:-

AUUGN Editor
PO Box 366
Kensington, NSW, 2033
AUSTRALIA

E-mail: auugn@munniari.oz.au

Phone: +61 2 543 3885
Fax: +61 2 543 5097

AUUGN Book Review Editor

The AUUGN Book Review Editor is Dave Newton (dave@teti.qhtours.oz.au).

Contributions

The Newsletter is published approximately every two months. The deadline for contributions for the next issue is Friday the 21st of February, 1992.

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 left at the top and bottom so that the AUUGN footers can be pasted on to the page. Small page numbers printed in the footer area would help.

Advertising

Advertisements for the AUUG are welcome. They must be submitted on an A4 page. No partial page advertisements will be accepted. Advertising rates are \$300 for the first A4 page, \$250 for a second page, 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 editor 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:

AUUGN Inc.
Back Issues Department
PO Box 366
Kensington, NSW, 2033
AUSTRALIA

Also please note that the prices for back issues published in AUUGN Vol 12 No 1 are incorrect.

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 Institutional Members as at 13/01/1992

(NSW) Department of Minerals & Energy	Communica Software Consultants
A.N.U.	Computer Power Group
AAII	Computer Science of Australia Pty Ltd
Adept Business Systems Pty Ltd	Computer Software Packages
Adept Software	Corinthian Engineering Pty Ltd
AIDC Ltd.	CSIRO
Alcatel Australia	Cyberscience Corporation Pty Ltd
Amdahl Pacific Services Pty Ltd	DMP Software Pty Ltd
Andersen Consulting	Data General Australia
ANSTO	Deakin University
ANZ Banking Group/Global Technical Services	Defence Service Homes
Apple Computer Australia	Department of Transport
Apscore International Pty Ltd	Dept. of Agricultural & Rural Affairs
Ausonics Pty Ltd	Dept. of Conservation & Environment
Australia Eds Pty Ltd	Dept. of Defence
Australian Airlines Limited	Dept. of Foreign Affairs & Trade
Australian Bureau of Agricultural and Resource Economics	Dept. Of The Premier & Cabinet
Australian Eagle Insurance Co. Ltd	Dept. of Treasury
Australian Electoral Commission	Duesburys Information Technology Pty Ltd
Australian Information Processing Centre Pty Ltd	ESRI Australia Pty Ltd
Australian Taxation Office	Eastek Pty Ltd
Australian Technology Resources (ACT)	EDS (Australia) Pty Ltd
Australian Wool Corporation	Emulex Australia Pty Ltd
Avid Systems Pty Ltd	Expert Solutions Australia
BHP CPD Research & Technology Centre	FGH Decision Support Systems Pty Ltd
BHP Minerals	Financial Network Services
BHP Research - Melbourne Laboratories	First State Computing
BICC Communications	Fremantle Port Authority
Bond University	Fujitsu Australia Ltd
Bain & Company	G. James Australia Pty Ltd
Ballarat Base Hospital	GEC Alsthom Australia
Burdett, Buckeridge & Young Ltd.	Genasys II Pty Ltd
Bureau of Meteorology	General Automation Pty Ltd
Byrne & Davidson Holdings Pty Ltd	George Moss Ltd
C.I.S.R.A.	Hamersley Iron Pty. Limited
Capricorn Coal Management Pty Ltd	Harris & Sutherland Pty Ltd
CITEC	Hermes Precisa Australia Pty Ltd
Codex Software Development Pty. Ltd.	Highland Logic Pty Ltd
Colonial Mutual	Honeywell Ltd
Com Tech Communications	IBM Australia Ltd
Commercial Dynamics	Iconix Pty Ltd
	Infonetics
	Information Technology Consultants
	Internode Systems Pty Ltd

AUUG Institutional Members as at 13/01/1992

Ipec Management Services
James Cook University of
 North Queensland
Labtam Australia Pty Ltd
Land Information Centre
Leeds & Northrup Australia Pty. Ltd
Macquarie University
Mayne Nickless Courier Systems
McDonnell Douglas Information
 Systems Pty Ltd
McIntosh Hamson Hoare Govett Ltd
Metal Trades Industry Association
Mincom Pty Ltd
Ministry of Consumer Affairs
Ministry of Housing & Construction (VIC)
Mitsui Computer Limited
Motorola Computer Systems
Multibase Pty Ltd
NEC Information Systems Australia Pty Ltd
Nucleus Business Systems
Office of the Director of
 Public Prosecutions
OPSM
Oracle Systems Australia Pty Ltd
Parliament House
Prime Computer
Public Works Department
Pulse Club Computers Pty Ltd
Q.H. Tours Limited
Queensland Department of Mines
Queensland University of Technology
Radio & Space Services
RMIT
SBC Dominguez Barry
SEQEB Control Centre
Signum Software Pty Ltd
Silicon Graphics Computer Systems
Snowy Mountains Hydro-electric Authority
Software Development International Pty Ltd
Sony (Australia) Pty Ltd
South Australian Lands Dept.
Sphere Systems Pty Ltd
St Vincent's Private Hospital
Stallion Technologies Pty Ltd
Stamp Duties Office
State Bank of NSW
Steedman Science and Engineering
Swinburne Institute of Technology
Sydney Ports Authority
Systems Union Pty Ltd
Tasmania Bank
Tattersall Sweep Consultation
Telecom Australia
Telecom Australia Corporate Customer
Telecom Network Engineering Computer
 Support Service
Telectronics Pty Ltd
The Anti-Cancer Council of Victoria
The Fulcrum Consulting Group
The Opus Group
The Roads and Traffic Authority
The University of Western Australia
Toshiba International Corporation Pty Ltd
Tower Technology Pty Ltd
TurboSoft Pty Ltd
UCCQ
Unidata Australia
Unisys
University of New South Wales
University of Queensland
University of South Australia
University of Sydney
University of Tasmania
University of Technology
UNIX System Laboratories
Unixpac Pty Ltd
Vicomp
VME Systems Pty Ltd
Wacher Pty Ltd
Wang Australia Pty Ltd
Water Board
Westfield Limited
Wyse Technology Pty Ltd

AUUG President's Report

Dear AUUG Member,

AUUG is at a crossroads. Having functioned successfully for a number of years as a small, but active, group of Unix technocrats whose main purpose was to stay in touch with technical developments and work in progress, today AUUG is trying to be all things to all people.

The last Executive Committee meeting agreed that we must change to reflect changing times, but without alienating our loyal members. Further, we must not only not alienate these people, but must deliver benefits to them even as we strive to deliver benefits to newer, more commercial (for lack of a better word) members.

I'd like to indulge in a couple of quotations that sort of sum up where we are with all of this:

"...an enterprise can't maintain itself today just by repeating what it did yesterday, if it doesn't grow it dries up, it is like something living, when it stops growing it starts dying..." (E.L. Doctorow, Billy Bathgate)

"If you want to make enemies, try to change something." (Woodrow Wilson, 1916)

AUUG has long tried to avoid a schism between the older and newer members, as happened in the US when Usenix broke away from what is now UniForum. We've taken the view that the market is too small to support two groups and, more to the point, as long as we focus correctly on the needs of all types of members, we should be able to remain unified.

We do acknowledge that we are not meeting our responsibility to our members in terms of delivering member benefits. We recognise that this is so, not due to lack of desire or good will, but because all the business of AUUG (other than managing membership which is handled by ACMS, the AUUG Secretariat) continues to be done by volunteers. It's also impossible to deliver any single benefit that meets the needs of all members, or that is consistently valuable to all members.

We are taking steps to rectify this. We've appointed our first actual AUUG staff member. Liz Fraumann commences working for AUUG two days per week from the beginning of February. Liz, who arrived in Australia from the US late last year with her husband, Roger, who is with UNIX International, has a wealth of experience both in the UNIX market and with user groups, trade shows, conferences, and the like. She will be responsible for the day to day business of AUUG, most particularly for providing a fast response to member enquiries and other correspondence.

We have also reappointed Symmetry Design to handle the promotion of AUUG, specifically related to the AUUG'92 conference and increased membership. There has been bitter debate among the Committee regarding the value of promotion. I firmly believe that either we grow or we die, and that the only way we can grow is if someone is actively promoting the organisation and working to attract new members. That is Symmetry's role.

I believe I was given the mandate to continue this activity at the Annual General Meeting last September, when we clearly spelled out what we had been doing in terms of promotion, publicity, membership drive, and so on. All members present seemed to concur that, while they personally might not be sympathetic towards marketing, promotion and the like, they acknowledged that we had to grow the organisation. They also seemed to acknowledge that the profile of new members was likely to be quite different than that of older members.

We need to know what you think, but we need specifics rather than vague criticism. I feel sometimes as

if there is a vocal minority that makes its views known, but that does not necessarily reflect the views of the membership as a whole.

The Committee is here to serve the members. We make decisions on the basis of what we believe to be right and best. We don't want to split, and we do want to serve the needs of all members. We all have to keep in mind that what may seem trivial or unnecessary to us may be of critical importance to others.

Please let us know what you want.

AUUG 1992 Summer Conference Series

This is a preliminary announcement and call for papers for the AUUG 1992 Summer Technical Conference Series.

The AUUG Summer Conference is a series of one day technical meetings held in regional centers around the country. The meetings not only attract local speakers, but also include invited interstate speakers.

The aim of the Summer Technical Conference Series is to supplement the annual AUUG winter conference by providing an informal, technical forum for the presentation and exchange of current work in the area of the Unix operating system. It is expected that the content of these meetings will provide technical issues which are relevant to programmers, systems administrators and experienced users.

1992 will be the third year that the Summer Technical Conference Series has been held. It will also be the first time that these meetings will held in all states and mainland territories.

Papers in all areas of Unix-related research and development are solicited for the programmes. Intending speakers should submit an abstract of their presentation. Papers selected for presentation will be published in the AUUG newsletter. Speakers may also be invited to present their papers at interstate meetings and at the 1992 Winter Conference in Melbourne.

Planning for the Conference Series has just begun. Dates of the regional meetings are not yet known, however they are expected to be held between February and April 1992.

Further information will be posted the the newsgroup `aus.auug` as it becomes available. Please direct any enquires to the Regional Organiser in your state:

City	Organsier	Company	Email	Phone
Perth	Alan Main	Functional Software	atm@pyrmania.oz.au.	(09) 4481204
Adelaide	Michael Wagner	Systems Services		(08) 212 2800
Melbourne	Ian Hoyle	BHP Research	ianh@resmel.bhp.com.au	(03) 560 7066
Hobart	Steve Bittinger	University of Tasmania	steveb@tasman.cc.utas.edu.au	(002) 20 2811
Canberra	Ross Hand	NEC Information Systems	rossh@spider.ento.csiro.au	(06) 246 4071
Sydney	Lucy Chubb	Softway	lucyc@softway.sw.oz.au	(02) 698 2322
Brisbane	Mark Addinall	Stallion Technologies	mark@stallion.oz.au	(07) 870 4999
Darwin	Phil Scott	Computer Science, NTU.	pscott@pandanus.ntu.edu.au	(089) 46 6519

or to the coordinator of the conference series:

Glenn Huxtable
University of Western Australia
glenn@cs.uwa.oz.au
(09) 380 2878

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,
Uniform,
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
Trosno	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
(02) 332 4066

Fax:

SESSPOOLE

SESSPOOLE is the South Eastern Suburbs Society for Programmers Or Other Local Enthusiasts. That's the South Eastern Suburbs of Melbourne, by the way.

SESSPOOLE is a group of programmers and friends who meet every six weeks or so for the purpose of discussing UNIX and open systems, drinking wines and ales (or fruit juices if alcohol is not their thing), and generally relaxing and socialising over dinner.

Anyone who subscribes to the aims of SESSPOOLE is welcome to attend SESSPOOLE meetings, even if they don't live or work in South Eastern Suburbs. The aims of SESSPOOLE are:

To promote knowledge and understanding of Open System; and to promote knowledge and understanding of Open Bottles.

SESSPOOLE is also the first Chapter of the AUUG to be formed, and its members were involved in the staging of the AUUG Summer'90 and Summer'91 Melbourne Meetings.

SESSPOOLE meetings are held in the Bistro of the Oakleigh Hotel, 1555 Dandenong Road, Oakleigh, starting at 6:30pm. Dates for the next few meetings are:

Wednesday, 4 March 1992

Thursday, 16 April 1992

Tuesday, 26 May 1992

Wednesday, 8 July 1992

Thursday, 20 August 1992

Hope we'll see you there!

To find out more about SESSPOOLE and SESSPOOLE activities, contact either **Stephen Prince** (ph. (03) 608-0911, e-mail: sp@labtam.oz.au) or **John Carey** (ph. (03) 587-1444, e-mail: john@labtam.oz.au), or look for announcements in the news-group **aus.auug**.

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, 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.au.oz

Technical enquiries to:

Frank Crawford (frank@atom.lhrl.oz) (02) 543 9404
or
Scott Merrilees (Sm@bhpese.oz) (049) 40 2132

Thank you

=====

1.2 Contact Details

Name: _____
Address: _____

Phone: _____
Fax: _____
E-Mail: _____

1.3 Site Details

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

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): < 1 1-10 10-100
 (not counting netnews) > 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:

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): < 1 1-10 10-100
 (not counting netnews) > 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 Yes No
 equivalent) is this acceptable?
- B8. Do you charge for connection? Yes No
 If yes, approximately how much (please
 also specify units, e.g. \$X/MB or flat fee)? _____
- B9. Any other restrictions (e.g. educational connections only).?
- B10. Additional Comments:

AUUGN BOOK REVIEWS

Open Systems: a business strategy for the 1990s

by Pamela Gray,
McGraw-Hill, London, 1991
(ISBN 0-07707244-8, hard-cover, 263pp, rrp. \$75.00)

reviewd by
Rolf Jester
Digital Equipment Corporation (Australia) Pty. Ltd.
(rolf.jester@sno.mts.dec.com)

The field of Open Systems is plagued by a great deal of nonsense, hype and self-serving part-truths. We are told by some that an Open System is any system that uses brand A of micro-processor, any system that uses operating system B, brand C database, or a particular networking standard. In fact in my twenty-odd years in this industry I have never seen greater confusion about fundamental and important issues. Sad to say vendors of hardware and software are chief among those creating the confusion. But few others, industry gurus, academics or writers of the media are doing much to help us get things straight.

Yet, while the debate about the merits of competing products and technologies will and must go on, it is in fact possible to achieve a high degree of clarity about Open Systems. In a free market there will inevitably be commercial conflict. But it is possible to articulate some basic definitions and standards that have consensus across the industry. And that is what users of information technology need - some solid foundational standards on which open systems can be built in such a way that they do not have to be torn down again at great cost later on.

The subject of Open Systems is about standards. That statement would be disputed by some people who would prefer that the product they happen to be pushing today be accepted as the unique manifestation of "Open Systems". If only all of us would do the right thing and buy their product then we would not have the problem of open systems at all. The reality, however, is that all of us have to live with multiple vendors' systems whether we like it or not. In fact many people do like it: recent AUUG-sponsored Australian research by DMR shows that the number of operating systems on the desk-top in each organisation is going to increase, not decrease. There will be more diversity in future. And the only way to make all those different pieces of hardware and software work together is to have agreed common standards. Those standards have to be at the interface level - between applications and operating system, between various software components, and between different systems. Standards have to be completely vendor-neutral and product-neutral.

And so Pamela Gray quite rightly devotes much of her book "Open Systems" to standards.

Although the topic of standards is complex, it can be made clear if taken a step at a time. Gray succeeds admirably in doing that, and thus renders an extremely valuable service to our industry right now. Many of the people I meet are understandably confused by the profusion of standards and would-be standards. They are not helped by the press statements of vendors pushing their particular view of life. But anyone who takes the time to read through Gray's book and follow her factual description of the standards environment will end up un-confused.

The book covers:

- Why standards are needed in the computer industry.
- What standards are needed.
- How standards are made - the bodies and the processes.
- The standards for the two major facets of open systems:

- portability and scalability,
- interoperability.

The key role of X/Open. Implications for users, with case studies.

Gray starts from a business perspective. She succeeds in presenting a clear view of the strategic implications of open systems in terms that are relevant to Information Systems management. The book is not aimed at general management, but it does achieve its aim of being readable by any interested reader. The case studies help a lot here. At the same time the book is firmly based on a sound understanding of the detail, and goes into enough detail to make a convincing case.

This is not a glib advertisement for any quick panacea. While Dr Gray backs her assertion that a "strategy based on open systems is the only logical way forward", she also takes a thorough look at the costs and the reasons why people have not yet implemented such systems. The case studies, for example, show the negative as well as the positive experiences.

Both aspects of Open Systems

There are two main aspects to Open Systems - portability and interoperability. This book is one of the few publications I have seen that manages to cover both aspects. Portability is important. Through UNIX and product-neutral standards like POSIX we are beginning to achieve the benefits of portability. But interoperability is equally important, because the newer standards-compliant UNIX systems must still work together with the enormous installed base of proprietary systems. Open Systems Interconnection (OSI) is one of the means of achieving that. One of the unfortunate facts of our industry is that the OSI people don't know much about UNIX, and the UNIX people don't know much about OSI. This book will but it will help both groups, and everyone else, see the total picture a bit better.

Coverage of standards bodies and the standards process

The popular trade press inevitably focuses only on the high-profile conflicts between competing consortia. The real standards are set by far less controversial bodies, the actual work being done by conscientious and dedicated engineers. Gray shows clearly the importance of the IEEE, ANSI, ISO, CCITT. She examines some key standards in a little detail, showing clearly for example, how IS 9945-1 (POSIX 1003.1) relates to vendor documents like AT&T's System V Interface Definition (SVID).

The author does a marvellous job of clarifying the relationship of standards and products. This is one of the most vexatious areas of this troublesome field, with particular software vendor's products being often equated to "standards". She describes some actual products, like AT&T USL's System V, the Open Software Foundation's OSF/1, XENIX and SCO Open Desktop, explaining how they relate to the standards.

X/Open

I agree with the author that "X/Open represents the greatest force for the eventual implementation of the many standards that the computer users need." Gray's coverage of X/Open is appropriately comprehensive, having a whole chapter devoted to it. We have not seen enough exposure for this truly international body in the Australian press, and it deserves far more attention from users in this country.

X/Open is a group that embraces all the largest computer manufacturers, the members of UNIX International and the members of the Open Software Foundation as well as both of those organisations themselves. The X/Open Portability Guide (XPG) set of standards are supported and adhered to by all major hardware vendors and therefore represent a real and much needed unifying force in the industry. These are not future standards yet to be created, but are published, readily available, mature and stable. Users and application developers who insist on XPG compliance from their hardware vendors are today able to achieve real vendor-independence and portability at the source level despite all the other differences between competing hardware and system software products.

X/Open is also useful because it does not add to the standards confusion, but adopts, publishes and promotes formal standards where they exist - like ISO 9945-1 (POSIX 1003.1). X/Open is a practical organisation and recognises that the current formal standards are not in themselves sufficient for creating a complete open systems environment. So, where no formal standards exist, as in the user interface area, it adopts a relevant de facto standard (the X Window System). If absolutely necessary it creates a new de facto standard by specifying the interface to an existing actual product in product-neutral terms. That was done with the ISAM specification for file management, for example, based on the Informix C-ISAM product. Gray does the industry and users a valuable service by highlighting the unifying role that X/Open plays.

Summary

"Open Systems: a business strategy for the 1990s" will have great value for IS Managers and other IS Professionals responsible for ensuring the long-term viability and productivity of their organisations' IS investments.

One of the things I would have liked to see improved is the Glossary; it is useful but could be much more comprehensive. There is a real need for a reference work giving a comprehensive explanation of the terms and abbreviations in this field. Although Open Systems is my daily bread and butter, I still find myself in need of such a work occasionally, and I would like to be able to refer enquirers to something.

The "Sources for further information" is good. I hope that many readers follow up and seek further information on standards promotion bodies like X/Open.

At \$75.00 recommended Australian retail price, the cost will unfortunately discourage many individuals and students, but this book is essential for all Information Systems professionals and managers. If you are at all concerned with or interested in Open Systems, buy it.

Rolf Jester
Secretary
Australian Open Systems Users Group.

AT&T is a trademark of the American Telephone and Telegraph Company.
C-ISAM is a trademark of Informix Software, Inc.
OSF and OSF/1 are trademarks of the Open Software Foundation.
SCO and Open Desktop are trademarks of Santa Cruz Operation, Inc.
UNIX, System V and SVID are trademarks of UNIX System Laboratories.
XENIX is a trademark of Microsoft Corp.
X/Open is a trademark of X/Open Company Limited.
X Window System is a trademark of Massachusetts Institute of Technology.

AUUG Book Club

Book Reviews

AUUG Inc and Prentice Hall Australia have formed the AUUG Book Club to give AUUG members a chance to obtain Prentice Hall books at a significant discount.

To obtain copies of the books reviewed here, fill in the order form that appears at the end of the book reviews. Don't forget to deduct 20% from the listed retail prices.

Review copies of these books were kindly provided by Prentice Hall.

If you would like to review books for further offers from the AUUG Book Club, please contact the AUUGN Book Review Editor (see page 5).

THE STANDARD C LIBRARY

by P.J. Plauger
Prentice Hall, RRP \$62.95

Reviewed by
David Newton
Q.H.Tours
<dave@teti.qhtours.oz.au>

Most experienced Unix programmers will have read something by J.L. Plauger which they would have found either useful or interesting. *THE STANDARD C LIBRARY* is a book which C programmers will find both useful and interesting, and a valuable reference for their library.

The most prominent aspect of the book is that it is not about programming in the C language but about the use and implementation of C libraries which makes it quite unique. It is expected the reader is already an experienced C programmers.

The stated purpose of the text is:

- To teach users and implementors *how the library was meant to be used and how it can be used.*
- To teach programmers *how to design and implement libraries in general.*

The next most prominent aspect of *THE STANDARD C LIBRARY* is that the ANSI/ISO standards for C libraries are reviewed and discussed in detail with the inclusion of 9000 lines of implemented code. In the words of the author; *seeing a realistic implementation of the standard C library can help you better understand how to use it.*

The code examples are written:

- To be as readable and exemplary as possible.
- To be highly portable across diverse computer architecture.
- To demonstrate a sensible tradeoff between accuracy, performance and size.

The structure of the book is simple. There is an introduction and a separate chapter for each of the fifteen headers which declare or define the entire standard C library. Each chapter includes portions of the ISO C standard and a discussion on background, customary usage, implementation, testing and problem both obvious and subtle.

P.J. Plauger played an important role in the development of the ANSI C standards. His insight into the standardization of C at the committee level gives a good deal of depth to the book. The other feature which makes it interesting is its historical perspective. Because C has been around for a while it has a heritage of historical reasons for why the C libraries exist as they now are in the standards. *The Standard C Library* gives the history of many of the library functions which is something you don't often get in language books. A good example is the floating point library and the difficulties faced in implementing hardware independent code in the face of increasingly diverse hardware architectures.

In conclusion the *The Standard C Library* is clear and concise, defining technical terms well and including good reference lists for further reading. The book includes the sort of practical information which comes from years of

experience and makes it a valuable resource for those interested in using or creating C libraries.

SOFTWARE ENGINEERING IN THE UNIX/C ENVIRONMENT

by William B. Frakes, Christopher J. Fox and
Brian A. Nejmeh
Prentice Hall, RRP \$39.95, 262 pages (paperback).

Reviewed by
David Carroll
Australian Nuclear Science and
Technology Organisation
<davidc@atom.ansto.gov.au>

I went to a course at King's Cross recently on Application Development in UNIX, and was rather disappointed by the whole thing. It was all Unix, and no application development, and apart from a small section on lex and yacc the most interesting thing about the three days was my naive wanderings through the Cross at lunch.

By one of those coincidences you're never supposed to write about I got this book to review a couple of days later. Coincidental, because it contains everything I was expecting from the course and didn't get.

That and the fact that it is one of the easiest textbooks to read I've seen in a while has meant it has given me a highly favourable impression. But, as the book itself does, lets look at some specifics.

It's basically a manual on how to write large and maintainable programs in C, and from within UNIX. Starting with the prerequisite introductions and definitions it then goes through the Software Design Process, showing you all the pretty diagrams and documents needed, and how (and why) to go about making such documents useful throughout the life of the project.

Then come Chapters 4 to 6, my favourite bit. This is the section which deals with C itself, and is divided into Program Readability, Low-level Programming and High-level Programming. As these titles suggest it moves from the overall perspectives of readability, functionality and maintainability right down to specifics about naming conventions, the use of macros, clear indenting, DeMorgan's law for Boolean

expressions and more. As an example of the authors' attitude, they maintain that clear code is of the utmost importance and should always be preferable over C's little tricks. The first rule of program optimisation, they say (though they spell it differently) is to avoid optimisation if possible. Then they show you where you should optimise, and how.

Continuing through the book we come to a look at the tools Unix provides for coding, making, archiving, debugging and several miscellaneous processes. Then they take a long look at program testing, debugging and maintaining (including a look at version control) and finally a very short look at future trends.

That's not quite it, because you also get (for no extra cost) five appendices. The first two concern the example used for application development, a small program called ccount which creates a useful metric for analysing C code (funny that). Complete project documents are given, from concept exploration to data dictionaries and dataflow diagrams and everything else (thirty two pages worth). Then complete source code, including the shell-script prototypes (another very useful feature) and the final C modules.

The last three appendices are simply a template for documents, a summarised list of good C programming practice, and a check-list for code inspection (not to mention a list of references and an index, but you probably could have guessed that).

The point about this sort of book is not, however, about how many nice features it's got, but how it's going to improve your own coding or system design. That is more or less up to you really, and as several people have pointed out, UNIX and UNIX programmers are not the most useful tools when it comes to the design and readability of large programs.

I do like this book, and hope it will allow me to circumvent some of my own bad habits. It's hard to imagine any improvements needed from the text itself to make it an easier task - but it still remains a significant challenge.

NETWORK COMPUTING SYSTEM TUTORIAL

by T. Lyons

Prentice Hall, RRP \$60.95, 334 pages (Paperback)

Reviewed by

Frank Crawford

Australian Supercomputing Technology

<frank@atom.ansto.gov.au>

This book describes the Network Computing System (NCS), which is a method of distributing software applications across a network. It is one of a series describing Apollo's Network Computing Architecture, the others being the protocol specifications, a programmers reference manual and a system administration manual.

Although the system described was originally based on the Apollo system, Hewlett Packard have now made it available for Domain/OS, HP-UX, SunOS, VAX/VMS and VAX/Ultix. In its simplest form it provides a Remote Procedure Call (*rpc*) interface similar to that from Sun Microsystems (*i.e. Sun RPC*) and now available on most systems (and the basis of *NFS*). However, at least as described in this book, it provides a number of other features such as network registries, network exception handling and facilities for *object oriented programming*.

The aim of this book, as the name implies, is to provide a tutorial for programmers using NCS. It makes no assumptions about any knowledge of networks or *rpcs* or of any particular operating system. Although NCS supplies versions both for C and Pascal and for various operating systems, the examples are based on the C version and UNIX.

The first chapter introduces the concepts behind Remote Procedure Calls, concentrating on the similarities between local (or normal) and remote procedure calls. It then goes into some of the basics of both the Network Computing Architecture and Network Computing System (which is an implementation of NCA).

The following chapter then goes on to give a simple example of *rpcs*, a program to query the network registries, or *location brokers*, to return the statistics that they maintain. This introduces some of the basic calls used by all *rpc* clients and the tools, such as the Network Interface Definition Language Compiler (*NIDL* Compiler) which converts the interface definition script into

a number of C header files. An example, a network messaging utility similar to *write*, is introduced which is then built upon throughout the rest of the book. This example is a client/server application, which ultimately includes a database to register server (*i.e. user*) information, administration utilities and multiple version support.

While building this application, topics such as how to catalogue programs with both the local and global location broker, the various data type and structures that are handled and error recovery techniques. It also serves as a reasonable example of how to build and maintain distributed applications.

The final chapter describes a number of features that implement some *object oriented* distributed computing features. Primarily this is based on the fact that the location brokers have the facility to register and retrieve information related to types, including the selection of appropriate *rpcs*.

This book is well written and easy to read and, as a tutorial, could be a very valuable addition. My only reservation is that it is really only relevant to Hewlett Packard's NCA. If you have an interest in such a system then this book is a must otherwise it is only of academic interest.

THE C PROGRAMMER'S COMPANION ANSI C Library Functions

by R.S. Jones

Prentice Hall, RRP \$49.95, 157 pages (Paperback).

Reviewed by

John Pollard

Australian Supercomputing Technology

<jpp@atom.ansto.gov.au>

From the back cover ... "This book is aimed at the C programmer who wants to know more about C library functions and understand their exact semantics.

Features include: * A discussion of the constants and types defined in the header files. * A chapter devoted to discussing each major category of library functions. * A number of examples illustrating how the library functions are used."

The functions described in each chapter are presented in alphabetic order to aid easy location and are given briefly like: "#include <ctype.h> int toupper(int c); toupper converts a lower-case character to an upper-case character ..." In all, the presentation is satisfactory with the main lack being the absence of an example for each function.

The book, considering it is meant for reference, reads well except for the annoying habit of the author of over using personalisation like, "I will now show you ...". Further, some examples do not function correctly. For example, a <math.h> example gives wrong answers because #define PI 3.1416 should be at least #define PI 3.1415927 to give answers correct in all figures printed and several examples give a compiler error because of the omission of #include <stdlib.h>

From the Introduction ... "Most C books focus on the 'language' aspects of C ... C library functions are discussed only in passing and in an incidental way ..."

The previous claim is an important consideration in the purchase of a 'companion' book. Inspection of ten recent books on C programming showed that the claim was true for seven of them. The other three contained as much as this 'companion' or more. One contained half as much again, mainly due to the numerous examples. An eleventh book inspected was entirely devoted to the run time library (for a particular PC C). It too was bigger than this 'companion' (and almost a quarter the price).

So where does the present book find a niche? Most probably for the experienced C programmer (as examples are rather few) who is intent on using strictly ANSI C (rather than a PC variant). If this is you, then the book is a handy size to keep in your brief case and is a reasonable 'companion' although somewhat over priced.

A BOOK OF OBJECT-ORIENTED KNOWLEDGE

by Brian Henderson-Sellers
Prentice Hall, RRP \$44.95, 297 pages (Paperback).

Reviewed by
Jagoda Crawford
Australian Nuclear Science and
Technology Organisation
<jc@atom.ansto.gov.au>

As the title suggests, this book *does* present the Object-Oriented Knowledge possessed by the author. The author is an Associate Professor in the School of Information Systems at the University of New South Wales. On reading the book it quickly becomes obvious that the author possesses a large deal of knowledge and information on the topic (10 pages of references are included together with a four page annotated bibliography). In fact the book can be seen as a survey on the topic. Looking at the references a number of papers have been published by the author himself; also aside from teaching at the University he has been involved in workshops. This knowledge and experience is seen in the text.

The book concentrates on the object-oriented approach to software engineering. It talks about analysis, design and implementation in general terms without presenting a particular programming language, together with a large number of references for further study.

The material is presented in five chapters, with a glossary of object-oriented terms provided and the references mentioned earlier.

In the first two chapters a historical perspective and an introduction to the object-oriented philosophy and terminology is given. Chapter three talks about software engineering, reusability, extendibility, reliability, *etc.* Views on object-oriented analysis and design are presented in chapter 4.

Chapter 5 is on implementation concepts: objects, classes, *etc.* What constitutes an object-oriented programming language is described together with a short survey of languages available at the writing of the book. A number of easy to follow examples are given when needed, to illustrate certain concepts or techniques.

The book is intended to be used as an introductory text to the object-oriented approach as well as providing course material for those involved in running training courses on the subject. This is achieved by providing the information in text form together with the course material, *i.e.* a summary of the text in point form as one will use on slides for courses. Each *foil*, or *exhibit*, occupies a full page which can easily be photocopied. This was the intention of the author, however, one problem is that the copyright statement is the usual one, stating that no parts of the book may be reproduced without written permission of the publisher.

I enjoyed reading the book, however I found that at times I would have liked more worked examples. The author wanted to introduce the concepts underlying the object-oriented paradigm without focusing on a particular programming language too early, as:

"..you may miss part of the overall paradigm by focusing too early on present-day language support."

My feeling was that I would have grasped the ideas quicker had I been studying a language concurrently with reading this book. On the other hand, I found the exhibits very useful in refreshing previously read information and they will not only be of use to course presenters but also students using the book as text.

In conclusion I think anyone that has just started or is about to start looking into object-oriented approach will benefit from obtaining this book in conjunction with some of the references it gives. An introduction on the object-oriented approach is given together with a vast number of references for further study.

AUUG BOOK CLUB & PRENTICE HALL AUSTRALIA

20% DISCOUNT TO AUUG MEMBERS

Please send me a copy/copies of the following books —

- Lyons/Network Computing System Tutorial**
RRP \$60.95* ISBN: 1361-7242-3 Paper 1991
- Jones/The C Programmer's Companion**
RRP \$49.95* ISBN: 1311-6948-3 Paper 1991
- Plauger/The Standard C Library**
RRP \$62.95* ISBN: 1313-1509-9 Paper 1991
- Frakes/Software Engineering in the UNIX/C Environment**
RRP \$39.95* ISBN: 1382-6496-1 Paper 1991
- Henderson-Sellers/A Book of Object-Oriented Knowledge**
RRP \$44.95* ISBN: 1305-9445-8 Paper 1991

*Deduct 20% from listed retail price (20% discount offer available until 28 February 1992).

Name: _____ Organisation: _____

Address: _____

(Street address only)

Telephone: _____

Please send my book/s on 30-day approval (tick box)


Enclosed cheque for \$ _____ (Payable to 'Prentice Hall Australia')

Please charge my: Bankcard Visa MasterCard

Credit Card No:

Expiry Date: _____ Signature: _____

Mail or fax completed order form to Prentice Hall Australia, PO Box 151, Brookvale NSW 2100

OR  Use our FAST PHONE SERVICE by calling Jacqui Long.
SYDNEY (02) 939 1333

A.C.N. 000 383 406



Prentice Hall Pty. Ltd.
7 Grosvenor Place, Brookvale NSW 2100.
Tel: (02) 939 1333 Fax: (02) 905 7934

A Paramount Communications Company

Multiprocessor Streams for Plan 9*

David Leo Presotto

AT&T Bell Laboratories
Murray Hill, New Jersey 07974
research!presotto
presotto@research.att.com

ABSTRACT

This paper describes an implementation of Streams for the Plan 9 kernel, a multi-threaded, multiprocessor kernel with a system call interface reminiscent of UNIX. Rather than port Dennis Ritchie's Streams to Plan 9, we changed the abstraction to fit more naturally into the new environment. The result is a mechanism that has similar performance and is internally easier to program.

1. Introduction

Plan 9 is a new computing environment being built and used by the Computing Science Research Center at AT&T Bell Laboratories. Plan 9 consists of terminals, CPU servers and file servers connected by various networks. These components run specialized operating systems based on a common multi-threaded kernel. The kernel runs on both uniprocessors and shared memory multiprocessors.

Plan 9 communicates via a number of different networks. Therefore we decided to base all our network code on a single structure. This allowed us to solve at once a number of problems such as flow control, memory allocation, and user interface. Given our past experience with it, we chose Dennis Ritchie's Stream I/O System [Rit84] to provide the structure for Plan 9. This coroutine-based design, introduced in the Eighth Edition, provides a clean, flexible mechanism for handling asynchronous I/O. Although Plan 9's kernel is unrelated to that of the Eighth Edition [McI85], the concept of Streams remained directly applicable. We have, however, made two major alterations.

Plan 9 runs on multiprocessor systems so we wanted to exploit their concurrency. In the Plan 9 kernel, the basic unit of concurrency is the process. We therefore converted Ritchie's coroutine-based design to a process-based one. As we shall see later, this change has both advantages and disadvantages.

Associated with the change to a process-based structure, we also had to reduce the number of threads. If we had made the most obvious change to convert each of Ritchie's coroutines into a process, we would have incurred very high CPU penalties. No matter how cheap we make our kernel processes they would never be as cheap as coroutines. Instead, we chose a structure that performs in one process what Streams does in many coroutines.

The result is a structure very similar to Streams but, we believe, easier to program. The interfaces, flow control, and memory allocation are the same. However, the freedom to allow processing modules to block and to use any resources available to a user process has made many pieces much easier to program. A process is a familiar programming construct.

In the rest of this paper we will refer to Ritchie's Streams simply as Streams and to Plan 9 Streams as Plan 9.

* This paper was originally published in Proc. of the Summer 1990 UKUUG Conf., London, July, 1990, pp. 11-19
Editor: Sunil K Das, City University London.

2. Data Structures

Plan 9, aside from minor changes, uses the same data structures as Streams. Our description here is very brief and is intended to highlight the differences. We refer the reader to Dennis's excellent BLTJ paper [Rit84] for a more comprehensive treatment.

The appendix contains the C definitions of our data structures.

2.1. Stream

A `Stream` is a full duplex channel connecting a device or pseudo-device to a user process. User processes insert and remove data at one end of the stream. Kernel processes acting on behalf of a device insert data at the other.

A stream is made up of a linear list of *processing modules*. Each module has both an upstream (toward the user) and a downstream (toward the device) *put procedure*. Data is inserted into a stream by calling the put procedure of the module at either end of the stream. Each module calls the succeeding one to send data up or down the stream.

2.2. Queue

An instance of a processing module is described by a pair of `Queues`, one for each direction. Each queue contains:

- a pointer to the put procedure
- a pointer to the next queue
- a linked list of queued data blocks
- the number of bytes queued
- the number of blocks queued
- a spin lock to control access to the data structure

Unlike a `Stream` queue ours has no *service procedure*. Also, since on a multiprocessor setting priority levels is not a valid synchronization mechanism, we require a spin lock from the operating system to control access to the queue.

2.3. Block

The objects passed through the stream are described by a data structure called a `Block`. Our blocks are identical to `Streams` and contain a *base pointer*, a *limit pointer*, a *read pointer*, and a *write pointer*. Each pointer refers to memory mapped into kernel space. The base and limit are never changed and are used to describe the data allocated to the block. The read and write pointers point to the start and end of usable data within the block.

There are two block types, *data* and *control*. Data blocks are used to pass information from process to process. Control blocks are used to control the action of the modules. They both have the same format. Data blocks are often queued in a processing module until some condition is met for passing them along or freeing them. Control blocks are never queued but are passed from module to module until one accepts and frees them.

Streams also have data and control blocks. However, their control blocks come in multiple flavors, all queueable; some with the same priority as data and some higher. Higher priority blocks are moved to the front of the queue. As a result, the routines used to manipulate these control blocks tend to be complex.

When a module's *put* is called it is passed a pointer to a block. If one desires to pass many blocks atomically, the blocks may be chained together and a pointer to the first is passed to the procedure. This is similar to the way *mbufs* are passed in the BSD kernel [Lef89]. Streams need no such concept since no two threads run simultaneously in a `Streams` procedure. UNIX System V `STREAMS` [Bac86] have a much more complicated construct to pass a multi-block message along with a single *put*. The System V construct is used both for atomicity (they originally had no other block delimiters) and for performance.

3. Algorithms

3.1. Memory Allocation

Stream memory is allocated at system start time. A list is kept for each of several fixed block sizes. A process that requests a size receives the smallest block that can hold the request. Synchronizing access of the free lists is performed using a spin lock per free list.

Since all stream code runs in the context of processes, whenever an allocation cannot be immediately processed the caller blocks until a block of the right size is freed. The result is that momentary surges in used blocks do not panic the kernel as they sometimes did in the Eight Edition.

The number of lists, the specific block sizes and the number allocated of each size depends on the kernel. Terminals tend to use more small blocks, the servers more large ones. The allocated blocks reflect this.

3.2. User Interface

A stream is represented at user level as a directory containing at least two files, `ctl` and `data`. The first process to open either file creates the stream automatically. The last process to close destroys the stream. Writing to the `data` file causes a switch to kernel mode. The process then copies the data into kernel data blocks and calls the put procedure of the first downstream processing module for each block. The last block of a write is flagged with a delimiter in the event that the downstream module cares about write boundaries. In most cases the first put procedure calls the second, the second calls the third, and so on until the data is output. Thus, data may often be sent without taking a context switch. A write lock at the top of the stream assures that no two processes can simultaneously insert data into the top of the same stream, which insures that all writes are atomic.

Our system has no `ioctl` system call. The syntax and semantics of `ioctl` in UNIX are so uncontrolled that we left it out. Writing to the `ctl` file takes the place of `ioctl`. Writing to the control file is the same as writing to a data file except that the blocks created are of type control. A processing module parses each control block put to it. The commands in the control blocks are simple ASCII strings. Therefore, there is no problem with byte ordering when one system is controlling streams in a name space implemented on another processor. The time to parse the control blocks is not important since the control operation is a rare one, usually used only when starting operation on a stream.

The stream system intercepts the control blocks that control configuration of the stream. These control blocks are:

`push name` to add an instance of the processing module `name` to the top of the stream.
`pop` to remove the top module of the stream.
`hangup` to send a control message containing the string "hangup" up the stream from the device end.

Other control blocks are read by each module they pass through.

Reading from the `data` file returns data queued at the top of the stream. The read terminates either when the read count is reached or when the end of a delimited block is read. There is a per stream read lock that ensures that only one process can read from the stream at a time. This ensures that the bytes read were contiguous bytes from the stream. Reading the `ctl` file is described in the section on multiplexing.

3.3. Device Input

When input exists at a device, the driver's interrupt routine wakes up a kernel process to carry the data upstream. A kernel process is an ordinary process with no user level segments allocated to it and is scheduled just like any other process. Message-based devices like Ethernet [Met80] may have many processes ready to carry the next message upstream so that many messages can be processed simultaneously.

The kernel process carries the message upstream through protocol modules. Eventually, the message is delivered to the most upstream queue. The kernel process leaves it there and wakes any user process blocked in a read on that stream. Thus, the only difference between input and output is that the user process must perform the copy of the data from kernel blocks to user space. This can be a benefit in our

multiprocessors in which each processor has a separate cache. If the kernel process were to copy the data into the user process it would most likely do it on the wrong processor and hence into the wrong cache. This would force the user process to fault it into its own cache, increasing the load on the shared memory bus.

3.4. Multiplexing

Most protocols need to multiplex several conversations onto a single physical device. This is added to our scheme using pseudo-devices, one for each multiplexed conversation. This is very similar to the way Eighth Edition handles TCP/IP. A group of pseudo-devices are coupled with a multiplexing processing module that is pushed onto a physical device stream. The device end modules on the pseudo-devices add the necessary header onto downstream messages and then put them to the module downstream of the multiplexor. The multiplexing module looks at each message moving up its stream and puts it to the correct pseudo-device stream after stripping whatever header it used to do the demultiplexing.

The user interface to a multiplexed protocol is a directory. The directory contains a `clone` file and a stream directory for each conversation. The stream directories are numbered *1* to *n* (see Figure 1). Opening the clone file is a macro for finding a free stream directory and opening its `ctl` file. Reading the control file returns the ASCII number of the conversation chosen. This allows the user process to find the corresponding `data` file.

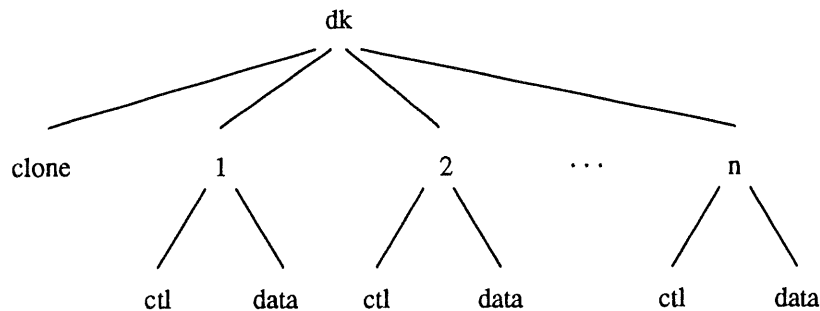


Figure 1

3.5. Pipes

Pipes, as in Eighth Edition, are just two streams joined at the device end. The `pipe` system call returns file descriptors for the data files of the two streams. The control files are inaccessible.

3.6. Helper Processes

Transport protocols need to retransmit lost data. However, to achieve true pipelining, the user process will want to queue data at the protocol module and return. Another process has to retransmit the data when needed since all put procedures must be called in the context of a process. For this purpose, processing modules can create kernel processes to perform such actions when needed. The processes are awakened on need by the processing module's put procedure or whenever a timer expires.

3.7. Flow Control

In any system that queues data one needs a mechanism to keep a queue with a slow reader from absorbing all of memory. We use a flow control mechanism similar to Streams. Each queue keeps a count of bytes and a count of blocks queued there. Whenever either exceeds a predetermined limit, the high water flag is set for that queue. Each caller of a processing module checks the high water flag for the next queue before calling its put procedure. If the next queue is past its high water mark, the would-be caller goes to sleep, leaving a pointer to itself in its queue (the rendezvous structure in `Queue`). When a process empties a queue past half its high water mark it wakes up any process waiting at the previous queue.

Modules implementing transport protocols with window schemes implement flow control a little

differently. Rather than go to sleep, they always pass data upstream. However, when the upstream queue is full, the transport module stops sending acknowledgements back to the remote system. Hence, the remote side will eventually stop sending. To open the window again, a helper process sleeps in the queue instead of the device process. When the next queue empties, the helper is awakened and it does whatever is needed to open the remote system's transmit window.

4. Performance

A different flavor of Streams cannot be evaluated without comparing it against earlier ones. Our results give a general idea of where the advantages and disadvantages of Plan 9 Streams lie. However, the systems compared are in many ways incomparable. The compilers, operating systems, and Streams code all have a considerable effect on the results.

For Plan 9 numbers we use 4 different configurations. Three are SGI Power Series machines with 1, 2, and 4 processors. We compare these against another 4 processor SGI Power Series running System V Release 3 and against a single processor MIPS M2000 system also running SVR3. The M2000 has the same CPU running at the same speed as the SGI machines. However, it has a considerably faster memory system. Outside of tight loops, this has a major impact on processing speed.

The other Plan 9 system is the Gnot terminal [Pik90]. This is a system developed in our center and now manufactured for us by AT&T. It uses a 25 MHz Motorola 68020. We compare it against a DEC MicroVAX 3. The machines on average are about the same speed. The Gnot CPU is about 4/3 the speed of the microVAX with a memory system that is about 2/3 the speed of the microVAX.

All tests measure both throughput and latency. The throughput is tested using the following program:

```
int i;
char buf[64*1024];
int p[2];

makeconnection(p);
switch(fork()){
case 0:
    close(p[1]);
    while(read(p[0], buf, sizeof buf) > 0)
        ;
    break;
default:
    close(p[0]);
    for(i = 0; i<ITER; i++){
        if(write(p[1], buf, sizeof buf) != sizeof buf){
            perror("write");
            exit(1);
        }
    }
    break;
}
```

The block size is chosen to be large to minimize the difference in system call speeds. The latency is tested using:


```

int p[2];
int i;
char c;

makeconnection(p);
switch(fork()){
case 0:
    close(p[1]);
    while(read(p[0], &c, 1) == 1)
        write(p[0], &c, 1);
    break;
default:
    close(p[0]);
    for(i = 0; i<ITER; i++){
        if(write(p[1], &c, 1) != 1){
            perror("write");
            exit(1);
        }
        if(read(p[1], &c, 1) != 1){
            perror("read");
            exit(1);
        }
    }
    break;
}
}

```

In both cases, we perform each operation for a large number of iterations to get an average time.

The first test (Table 1) compares Plan 9 pipes against SVR3 normal pipes, SVR3 stream pipes, a BSD sockets implementation running under SVR3, and Tenth Edition Streams. Since only two processes are involved in all configurations and no processing is being performed by processing modules, we are comparing the speed of the basic plumbing in the systems.

Table 1 - Pipes		
system	throughput MBytes/sec	latency ms
SGI/1 CPU Plan 9	6.0	.29
SGI/2 CPUs Plan 9	8.4	.21
SGI/4 CPUs Plan 9	8.4	.28
SGI/4 CPUs sVr3 old pipes	4.5	.51
M2000 sVr3 stream	8.0	.51
M2000 sVr3 sockets	8.0	.36
68020 Gnot Plan 9	1.79	1.67
uVAX 3 10th Edition spipe	1.04	1.69

From Table 1 we can see that for the large machines, Plan 9 has lower latency. This was the expected result since the straight call structure requires many fewer instructions than traditional Streams which must

schedule each service procedure in addition to calling its put procedure. The dip of .08 ms when going from 1 processor to 2 is the result of concurrency. The second process is starting up before the first has returned from queuing its block.

The unexpected result is the rise from .21 ms to .28 when going from 2 processors to 4. We believe that this is contention over the process queue. We hope to verify this assumption before this paper is presented.

The low single processor SGI throughput for Plan 9 compared to the M2000 reflects the slower memory on the SGI box. When we use multiple processors, we take advantage of the concurrency and our throughput passes all the others.

Plan 9 on the Gnot compared to the Tenth Edition on the MicroVAX is less impressive. We still have a definite advantage in throughput. However, given the ratio of machine speeds, we should be much better in latency. Profiling the kernel showed that the disappointing latency time was due entirely to the MMU. The MMU on the Gnot only retains one process state. Whenever we switch context we do a lot of faulting to refill the MMU. The reason the throughput doesn't suffer from this problem is that the pipelining causes a lot of data to be moved per context switch.

To compare the performance of kernel driver processors to performing puts at interrupt level, we used our most prevalent network, Datakit [Che80]. It is both a local and wide area network spanning all of AT&T. The MicroVAX, SGI, and M2000 each have 8 megabit/sec links to Datakit. However, due to constraints in the Datakit the highest throughput is 2.6 megabits. The Gnots have a slower 2 megabit link [Pre88]. Table 2 presents performance of various systems through the Datakit. In all cases the remote system is a Plan 9 SGI processor. Once again, Plan 9 throughput matches or exceeds the throughput of the other systems. However, latency is worse. This is the price paid for using kernel processes to send device data upstream rather than doing it in the interrupt routine. The degradation is especially evident in the Gnot since it is the worst at process switching.

Table 2 - URP/Datakit		
system	throughput KBytes/sec	latency ms
SGI/2 CPUs Plan 9	235	1.4
SGI/4 CPUs Plan 9	235	1.4
M2000 sVr3	235	1.2
68020 Gnot Plan 9	100	5.8
uVAX 3 10th Edition	85	3.2

Finally, we present some Ethernet performance results. We don't compare these against other systems since the protocol we use, Nonet, currently runs only on Plan 9. It was designed as a low weight transport protocol. It should be noted that the throughput figures are higher than any we've seen published to date for an Ethernet. This is as much a function the protocol as it is of Plan 9.

Table 3 - Nonet/Ethernet		
system	throughput KBytes/sec	latency ms
SGI/2 CPUs Plan 9	950	1.4
SGI/4 CPUs Plan 9	950	1.4

Related Work

We must mention Larry Peterson's *x*-Kernel work [Pet89]. The *x*-Kernel is very similar to Plan 9 Streams. It is used primarily to study the decomposition of network protocols. The process structure, multiplexing, and data structures are virtually identical to Plan 9 Streams.

The greatest differences between our systems are:

- 1) The *x*-Kernel uses very light weight kernel processes. They are just a PC and a stack. Our kernel processes carry all the baggage of user processes.
- 2) Rather than queue data at the user process and wait for the user process to read it, *x*-Kernel kernel processes call a put procedure in the user's address space which moves the data directly into user memory.
- 3) The message in the *x*-Kernel is in the form of a tree of blocks. A pointer to the top of the tree is passed through the processing modules. We use a linear structure.

Published performance of the *x*-Kernel is similar to ours with lower latency times. We hope to borrow some of the ideas of the *x*-Kernel to improve our own performance.

Conclusions

We have presented another variation of streams. The main advantage to Plan 9 Streams is making use of concurrency in multiprocessors. We have a very subjective belief that the process based model is easier to program than the coroutine based one.

The performance results show that the Plan 9 model has high throughput. However, the contexts switches caused by the kernel processes increase latency. Further research and tuning is required to reduce these costs.

5. References

- Bac86. M. Bach, *The Design of the UNIX Operating System*, Prentice-Hall (1986).
- Che80. G. L. Chesson and A. G. Fraser, "Datakit Network Architecture," in *IEEE Comcon '80* (January, 1980).
- Lef89. S. Leffler, M. K. McKusick, M. Karels, and J. Quarterman, *4.3BSD UNIX Operating System*, Addison Wesley (1989).
- McI85. M. D. McIlroy, *Unix Programmer's Manual, Eighth Edition*, Bell Laboratories, Murray Hill, NJ, USA (February, 1985).
- Met80. R. Metcalfe, D. Boggs, C. Crane, E. Taft, J. Shoch, and J. Hupp, "The Ethernet Local Network: Three Reports," CSL-80-2, XEROX Palo Alto Research Centers (February, 1980).
- Pet89. L. Peterson, "RPC in the X-Kernel: Evaluating New Design Techniques," in *Proceedings Twelfth Symposium on Operating Systems Principles*, Litchfield Park, AZ (December, 1989).
- Pik90. R. Pike, D. Presotto, K. Thompson, and H. Trickey, "Plan 9 from Bell Labs," in *UKUUG Proceedings of the Summer 1990 Conference*, London, England (July, 1990).
- Pre88. D. Presotto, "Plan 9 from Bell Labs - The Network," in *EUUG Proceedings of the Spring 1988*

Conference, London, England (April, 1988).

Rit84. D. M. Ritchie, "A Stream Input-Output System," *AT&T Bell Laboratories Technical Journal* 63(8) (October, 1984).

UNIX is a registered trademark of AT&T Bell Laboratories

Datakit is a registered trademark of AT&T

Appendix

```
/*
 * operations available to a queue
 */
typedef struct Qinfo    Qinfo;
struct Qinfo
{
    void (*input)(Queue*, Block*); /* input routine */
    void (*oput)(Queue*, Block*); /* output routine */
    void (*open)(Queue*, Stream*);
    void (*close)(Queue*);
    char *name;
};

/*
 * We reference kernel memory via descriptors kept in host memory
 */
typedef struct Block    Block;
struct Block
{
    Block *next;
    uchar *rptr; /* first not consumed byte */
    uchar *wptr; /* first empty byte */
    uchar *lim; /* 1 past the end of the buffer */
    uchar *base; /* start of the buffer */
    uchar flags;
    uchar type;
};

/* flag bits */
#define S_DELIM 0x80 /* this block is the end of a higher level message */
#define S_CLASS 0x07

/* type values */
#define M_DATA 0
#define M_CTL 1

/*
 * a list of blocks
 */
typedef struct Blist    Blist;
struct Blist {
    Lock;
    Block *first; /* first data block */
    Block *last; /* last data block */
    long len; /* length of list in bytes */
};
```

```

/*
 * a queue of blocks
 */
typedef struct Queue Queue;
struct Queue {
    Blist;
    int nb; /* number of blocks in queue */
    int flag;
    Qinfo *info; /* line discipline definition */
    Queue *other; /* opposite direction, same line discipline */
    Queue *next; /* next queue in the stream */
    void (*put)(Queue*, Block*);
    Rendez r; /* flow control rendezvous point */
    void *ptr; /* private info for the queue */
};
#define QHUNGUP 0x1 /* flag bit meaning the stream has been hung up */
#define QINUSE 0x2
#define QHIWAT 0x4 /* queue has gone past the high water mark */

/*
 * a stream head
 */
struct Stream {
    Lock; /* structure lock */
    int inuse; /* use count */
    int hread; /* number of reads after hangup */
    int type; /* correlation with Chan */
    int dev; /* ... */
    int id; /* ... */
    QLock rdlock; /* read lock */
    QLock wrlock; /* write lock */
    Queue *procq; /* write queue at process end */
    Queue *devq; /* read queue at device end */
};
#define RD(q) ((q)->other < (q) ? (q->other) : q)
#define WR(q) ((q)->other > (q) ? (q->other) : q)
#define PUTNEXT(q,b) (*(q)->next->put)((q)->next, b)
#define BLEN(b) ((b)->wptr - (b)->rptr)
#define QFULL(q) ((q)->flag & QHIWAT)
#define FLOWCTL(q) { if(QFULL(q)) flowctl(q); }

```

A Scientific Visualization Tool

Jack Dikian

**Silicon Graphics Computer Systems
446 Victoria Road,
Gladesville NSW, 2111**

(+61 2 879-9500)

jack@syd.sgi.oz.au

ABSTRACT

Scientific visualization is a technique for representing raw numerical data in visual images that model the interactions of objects and forces in the real system from which the data originally came.

One of the main purposes of scientific visualization is to aid scientists in understanding vast amounts of data generated by computers and other equipment. These images can be manipulated on screen as if they were actual physical systems. An image can show on the screen, in a few minutes, the effect of a new or changed variable on a system. To analyze the same effect from several thousand numbers covering reams of computer paper takes considerably longer, and lacks the completeness and immediacy of the visual model. Scientific visualization, however, also requires the use of very specialized software and in some cases the collaboration of specialist computer engineers.

Many large workstation vendors such as Silicon Graphics Inc, Stardent Computer Inc¹ and IBM are now making available, generic interactive visualization environments that give users access to graphics, visualization and imaging modules without the associated programming effort. This article takes a look at the Silicon Graphics solution, an application development environment for end-users and software developers called IRIS Explorer.

1. INTRODUCTION

Visual processing gives us the ability to produce realistic, three-dimensional, colour images and interact with them in real-time; just as we might manipulate an object held in our hand. We can absorb information and explore ideas in ways previously thought impossible. Visual processing has already transformed the way we work and interact with information. Future developments promise to make information analysis and synthesis even more intuitive and immediate.

Once a visual representation is generated; a bone fragment for example. An anthropologist working in the field can photograph the bone, scan the image, reconstruct a virtual model and begin to test

1. Stardent Computer Inc announced it would form a new corporation - AVS Inc. The new company, comprised of 25 former Stardent employees will operate as a separate entity.

hypotheses, extrapolate an effect over a period of time, apply new conditions or variables, or invent a new system all together. Today, scientific visualization is heavily used in astrophysics, meteorology, mathematics, molecular modeling, particle research, and medical imaging amongst many others.

Transforming data from the initial raw form to a display point, however, requires significant processing. Images are processed by the use of a number of operations including, correction and reduction of noise and distortion, polynomial wrapping, contrast enhancement by light-source shading, use of colour to show contours or group data in a given range².

Scientists in many of the fields mentioned above often do not wish to take on the difficult task of coding these operations. Also, many scientists use commercial software packages and wish to extend the systems with their own algorithms and techniques. Up to now, specialist scientists have relied on the services of computing professionals, and/or the use of purpose built commercial packages to facilitate the generation of computer models. There is a growing need therefore for systems that provide rich visualization development environments designed to be used by both programmers and non-programmers alike.

Systems such as Stardent Computer Inc's AVS³ (Application Visualizer System), IBM's newly announced Visualization Data Explorer/6000 and Silicon Graphics Inc's IRIS Explorer are here to fill that gap. Both the Stardent and the Silicon Graphics systems are quickly becoming industry standard environments.

2. IRIS Explorer

The IRIS Explorer which is bundled with the latest release of the Silicon Graphics IRIX operating system is a scientific visualization package that lets you read in, analyze, manipulate, and render complex data sets without the need for programming. Each step of the data input and output, analysis, and rendering is performed by a module that you place into a "network". The network is configurable on the fly using a simple point and click interface. A connect-the-modules approach invites users to integrate data and applications by visually connecting software modules into flow-chart maps or networks.

The strength of the system, besides its ease of use, X-window and Motif compliance, lies in its rich set of supplied modules. An Explorer module performs a specific action on the data that passes through it. Each module accepts data, acts on it in some way, and outputs the result to the next module downstream. Explorer modules are in essence made up of two parts. The first is the visible portion or control panel that contains the input/output ports as well as parameters, the values of which can be set and altered by widgets. The internal portion of a module contains the actual computational algorithm (C, C++, Fortran) and the module wrapper or module interface. Modules may be turned off, or disabled; and can be "fired" (executed) on a machine other than the one where the network is running. A large number of modules provide a wide variety of functions including file I/O, data analysis and of course visualization. Modules such as "BlendImg" which computes a blended image, "BlurImg" to blur an image, "Contour", "ForwardFFTImg", "FourierCrossCorrImg", "Rendercr" and many many (over 150) others are held in what Explorer calls the Module Librarian. The non-programmer can develop a specific visualization environment in order to analyze, and render his or her data by wiring together these pre-

2. Newman & Sproull Principles of Interactive Computer Graphics McGraw-Hill, Second Edition. 1979

3. North Carolina Supercomputing Center officially become the international AVS Center. This center will gather, standardize, catalog and maintain a collection of AVS modules currently available, along with modules created by users for the public domain.

defined modules. A programmer can modify, or add to the library of modules by writing C, C++, or Fortran code using Explorer's Module Builder.

The Explorer network is created by the Explorer Map Editor. The map editor is the work area in which modules are assembled and organized into an operational network. The whole thing is really very much like designing an application flow chart graphically, but with the one very large difference in that you can place real data at one end, and real results will appear at the other. The map editor allows the user to create, open, edit, save, and delete networks, create, disable, enable and destroy modules, run the network, and incorporate the network into an application.

When one draws a flow chart of an application, however, one would normally depict the flow of control rather than the flow of data. To say that developing applications in Explorer is as easy as drawing flow charts, therefore, needs further explanation. When drawing flow charts, we sometimes include process blocks or modules that have the task of converting data from one module to the next. In general however, flow charts do not describe data conversion, and an assumption is made that data conversion will be handled in the physical model viz the development phase. Explorer networks on the other hand are alive, and there is no such thing as a distinct development phase. IRIS Explorer is a strongly typed system. Two modules can only be connected if they share the same data types. The IRIS Explorer Module Builder helps the user define the external attributes of a module such as which data types the module will accept on its input port and the data produced on its output port. The Module Builder can also generate the code needed to interface from the outermost layer of the module (the visible portion), to the computational function, where the module's algorithm resides.

The need for data typing arises in several stages of development and execution. For example, when trying to wire two modules together, the map editor ensures that the two ports have compatible data types. Also, when a data object crosses a machine boundary, some component of the system needs to know the details of its internal structure so that the correct data representation conversion can be performed. IRIS Explorer has five major data types which are aggregates of other more primitive types such as ints, strings, floats *etc.* The five main types are:- Lattice, Parameter, Pyramid, Geometry, and Unknown. This data abstraction allows any one of the above data types to represent an entire class of data, on one hand, and a specific instantiation, on the other. For example, in its most general form, the lattice (essentially a multidimensional array) data type can represent any multidimensional array, whether byte structured, floating point, integer, short, or double precision. A specific instance, however, can represent only a two-dimensional lattice in byte format.

Two major Explorer utilities are intimately tied to the above data types. These are the Explorer Module Builder, and DataScribe, the data conversion utility.

The Explorer Module Builder is a graphical user interface tool that lets the user modify existing modules, and build new ones. The module builder automates the module building process so that the typical module requires no programming beyond that needed to write the computational function. The supplied port and parameter information is used by the module builder to automatically generate code, in the form of the Module Data Wrapper, to provide an interface between the outermost layer of the module, to the computational function. The data wrapper decomposes Explorer's data structures into a form usable by the user written function. For example, it can be arranged such that there is a relationship between the calling sequence of the computational functions, the function arguments and the port data types.

IRIS Explorer also comes complete with a data converter (DataScribe) which allows explorer to read a multitude of new data-set file formats. DataScribe can be used to convert data from an external source, such as an application or disk file, into explorer datatypes. It can also be used to convert explorer datatypes into data that can be used by other applications.

Internal data conversion also helps make this system suitable in a heterogeneous environment. Any

number of modules can be setup such that they execute on remote hosts, including the Cray. Data between these machines is transferred seamlessly. The distributed execution architecture allows a scientist working in computational fluid dynamics for example, run his mesh generation and post processing on an IRIS workstation while running his flow solver on a Cray.

As a very simple example of how you might use Explorer, let's say you have an image file created by saving a part of the screen. You may wish to enhance the contrast of the image, before displaying it again. A traditional method may be to write an application that reads the image file, applies a histogram equalization technique which rescales the data values of the image such that the cumulative histogram of the output image is approximately linear, before writing more code to display the image in the correct aspect and size. As simple as this application may seem, a large amount of knowledge is assumed. The format of the image file for example, contrast enhancing methods, the availability of underlying graphics libraries, as well as a working knowledge of some programming language. To perform this using Explorer takes just a few minutes. The steps are:

- Retrieve the Read Image module, the Histogram Equalization module, and the Display Image module from the Explorer library. Place these into the Explorer Map Editor.
- Wire the output of the Read Image module into the input of the Histogram Equalization Module.
- Wire the output of the Histogram Equalization module into the input of the Display Image module.

Now you have created an Explorer network. To run this network, all that is left to do is use the widget-based input parameter in the Read Image module to nominate the name of the image file you wish to read. The Display Image module also provides extra functionality. For example, it allows the user to pan around the image by using the mouse, as well as providing a zoom dial. This is a very simple Explorer network. There is no limit to the number and ways modules can be wired together.

3. Conclusion

IRIS Explorer is therefore a complete application creation system and user environment that provides visualization and analysis features for computational scientists and engineers. It is useful for those whose needs are not met by commercial software packages, and or, those who wish to extend existing systems with their own functions. IRIS Explorer runs across the entire Silicon Graphics product range. This product gives us a whole new paradigm for using machines in a heterogeneous environment. Modules within a map can be executed on connected machines. Modules and maps are available to cover functions as diverse as computational fluid dynamics, Earth Sciences, Molecular Modeling, and Medical imaging. And all this for free...?

Jeeves, the Butler

by Bernd Felsche
(bernie@DIALix.oz.au)
Copyright 1990,1991
MetaPro Systems Pty Ltd
Victoria Park, Western Australia

The butler is a daemon which acts as a general service agent for users. Initially intended to work around a 'feature' of an application package, the butler is also capable of other tasks, the scope of which may be extended by adding commands to its library. One attraction to the use of the butler, is the syntax of the command which instructs the butler as to what to do.

Butler architecture lends itself to privileged tasks, and in fact, several butlers, addressed by different names, could be used. Implementation, limitations, extensions and source code will be discussed in detail.

1. DEALING WITH FRANKENSTEIN

The butler was developed to provide a workaround for a database server which would hold a tty open, after launching the server as a daemon in background. This presented a problem because the terminal windowing environment refuses to close the window if it is open by any process, thereby preventing the user from logging out, having started a server.

Prior to the butler, all servers were started on system restarts, and after backups. This imposed a heavy load on system resources, consuming process table slots, shared memory segments, physical RAM and swap space. Furthermore, there was difficulty in starting a server manually, if a new one was required, as it required special tricks to prevent it hanging windows, or indeed terminals. It was decided that a service agent should be provided, to allow users to remote-start servers on demand, and to shut them down subsequently.

Rather than dive in and solve the specific problem, an abstraction, allowing for the service agent to perform a variety of predetermined tasks was adopted. The term "butler" was coined, providing a useful, everyday (?) concept which could be associated with its place in the system.

2. FRANKENSTEIN MEETS JEEVES

A simple, natural command syntax was required as all users do not share our ideals of UNIX terseness.

To further exploit the pun on a butler, and to enforce a simple command syntax, the command is called `jeeves, .`

The trailing comma is important, as it enforces the concept of having somebody else cope with a problem, and is more natural in english syntax. In fact, some commands appear almost like plain english, probably leading to accusations of heresy. It would not be unusual to see commands like "`jeeves, start golf with holes=19`" (a contrived example).

For the curious, a user entering `jeeves,` by itself is greeted with an interrogatory `Yes?` prompt.

`Jeeves,` the user interface, is a minimal program for reasons which will be explained later.

3. JEEVES TALKS TO DAEMONS

A method of isolating the user interface from the application was required, and the named pipe, aka FIFO, was selected.

FIFOs already serve similar tasks in System V `lp` spoolers and in `cron` tasks. The main advantage over other System V inter-process communications (IPC) is in the almost total isolation between the communicating processes. Only the information in the pipe, and the open/close state are known to the process at the other end of the pipe. Of course, the FIFO is also the most elegant way to perform IPC at shell script level.

`Jeeves`, channels user requests via a known FIFO, to the butler, which is actually the daemon, reads instructions from the FIFO, and then attempts to execute them in a limited environment.

Here is a code fragment to implement `jeeves`:

```
trap 'echo "no butler in residence"; exit 1' 15

(sleep $PAUSE ; kill -15 $$) 2>/dev/null &
echo "$request" >${BFIFO}
kill $!
```

Note the mechanism employed to ensure that `jeeves`, doesn't block indefinitely if there is no reader at the other end of the FIFO.

Although the butler has only three built-in commands (it could be as few as one, but the other two are trivial to implement anyway), its vocabulary is extended by adding commands to a special library. Only built-ins, and those commands in the library will ever be executed.

Due to the butler operating in a sterile environment, there is no way of spoofing it by users setting strange `PATH` or `IFS` environment variables. Other means of attack are still possible, but will require special privileges.

To execute non-built-in commands, the butler employs a background process, as there must be no assumption about how long external commands might take. This leads to a minor complication to prevent "zombies".

All the "real work" done by the butler is in a single `while` loop, which follows.

```
# read commands from the pipe and process them in subshells.
wtail < ${BFIFO} | while read cmd args ; do
  command='echo $cmd | tr "[A-Z]" "[a-z]"`
  case $command in
    the_butler_did_it)
      break ;;
    silent)
      : ;;
    hello)
      echo "$name: HELLO `date` $BSPPOOL/${BFIFO}" ;;
    *)
      ( but.servant $command $args )& ;;
  esac
  trap 'trap 15' 15 # wait for zombies
  (sleep 1 ; kill -15 $$ 2>/dev/null) &
  wait
  kill $! 2>/dev/null
done
```

Note the three built-in commands defined in the `case` statement. The first, causes the butler to exit. It is a deliberate attempt to discourage users from shutting it down too often.

Again, a shell `trap` is used, this time to stop the butler waiting too long for background processes to finish.

4. YOU JUST CAN'T GET GOOD HELP THESE DAYS

The `but.servant` you may have noticed in the previous code fragment reveals a further allusion to the butler not actually doing anything other than being there.

`but.servant` is the servant employed to run each external command asynchronously. A clear advantage to this is that the butler will always initiate commands in a reasonably predictable way, no matter how much the commands library is extended.

The background task figures out if the command even exists, and does various checks along the way.

```
command=`basename $1`
shift
args="$*"

cd $BLIB

if [ ! -f $command ] ; then
    echo "$usage0ommand $command not found" 1>&2
    exit 3
fi
if [ ! -x $command ] ; then
    echo "$usage0ommand $command not executeable" 1>&2
    exit 3
fi
exec ./ $command $args
```

The use of `exec` saves a process fork and preserves the child-parent relationship between the butler and the command which the user wants.

5. COPING WITH THE END OF THE WORLD

Not surprisingly, we like to have a butler around all the time, so we have one kicked off by `init`, set to a user id of `butler`. This helps to manage the workload on the butler, and stops it from running too many processes.

A side effect of running a daemon from `init` is that if there is some condition causing the daemon to exit prematurely, `init` will refuse to respawn it if it has to do so several times a minute.

In the situation of the butler, this could happen for example if the working, spool directory doesn't exist, or if it can't find a FIFO to listen to. Under these circumstances, instead of `exiting`, the butler simply goes to sleep, or to be more precise, becomes comatose, awaiting a kill signal.

This allows the system administrator to re-establish system sanity, without having to fight `init`'s inane messages.

6. ABBERATIONS AND DISCOVERIES

While developing butler, several discoveries were made about some System V features which haven't been documented anywhere.

The round-robin UNIX scheduler exhibits unexpected behaviour when multiple tasks are waiting on the same event. This was discovered during early development, when the feasibility was being explored.

Ten asynchronous processes were instructed to write to the same FIFO, which had not been opened for reading by any other process. When all ten had blocked, waiting for a reader, a reader was started, revealing that the ten waiting processes were run in reverse order, i.e. the last ran first, etc. It took some research, and some Usenet news traffic to resolve why this happens.

In effect, processes waiting for the same event get placed into a queue, which is then traversed in the last to first direction, when the event occurs. In effect, it's a stack.

It results in the FIFO apparently becoming a LIFO. The butler has no way of dealing with this situation.

Some manufacturers have tampered with the scheduler, so it may not exhibit this behaviour on your machine. A quick test is to execute the following shell commands and watch the results.

```
$ cd; mknod FIFO p
$ for i in 1 2 3 4 5 6 7 8 9 0 ; do echo $$ >FIFO & ; done
[ ... background process id's not shown ... ]
$ while read pid ; do echo $pid ; done < FIFO
[ ... process list ... ]
```

The process list will be in random order on a loaded system, and will probably be in reverse order otherwise.

The second aberration is that a shell script's \$0 is null if it is kicked off by `init`. This may be a shell bug, and I haven't yet received a satisfactory explanation. It appears that the shell cannot resolve its name when standard input is closed.

The butler copes with the situation by having default names built into scripts. It's by no means a satisfactory solution because the software has to be modified if more than one butler is required per system.

Finally, FIFOs are slightly strange creatures if you expect them to behave like files. The more observant reader will have noticed the `wtail` command in the butler, which feeds the `while` loop.

All standard UNIX commands will exit if the FIFO from which they are reading is closed by all processes writing to it. This is also true of shell loops and the `tail(1)` command.

An initial work-around was to place the `while` loop within another, which looped infinitely. This solution was shown to be unsatisfactory, as a subshell is created for each inner loop. The cost, in computing load, of starting a process is quite large. Jeeves, believed too often, that there was no butler present, because there was no FIFO reader for a few seconds.

`wtail` is a simple C program that keeps trying to read its standard input, and then places what it gets, on standard output. End-of-file is effectively ignored. Reading from a FIFO, with no writers, does not

block. Only an open(2) blocks.

7. LOCK UP YOUR BUTLERS

The most obvious enhancement to the butler is to implement a privileged one. To do this, the only major changes would be to make the front-end setuid, and to restrict access to the FIFO.

By prepending the user name to the command sent via the FIFO, the butler can ascertain if it should execute a particular command.

Built-in commands like the `do-nothing` `silent` require no restrictions, whereas `the_butler_did_it` should be restricted to a few responsible users. An access control list could be either hard-coded, and/or provided externally.

Access control lists for each external command provide a large degree of control over who should be able to access a particular external command. As the butler already gets a servant to run each external command, there will be little impact on the rate at which the butler handles requests.

Security violation attempts could be selectively mailed to prevent further attempts.

Several butlers can be run with different userids, performing a variety of functions for "ordinary" users, without necessitating the sharing of privileged account passwords.

8. BUGS WHICH HAVEN'T BITTEN

As long as commands remain short, the mechanism of submitting an entire request via a FIFO is fairly safe. However, it is only a matter of time before `jeeves`, falls in the buffered I/O pit.

Most UNIX commands buffer their I/O and try to read/write fairly large chunks at a time. If you were to try to submit a large request via `jeeves`, it may be split over two write system calls. This can lead to disaster, if another process writes to the FIFO between your request's writes, then the command will at best, be garbled.

There are two basic ways in which this problem could be averted. Keep the amount written to the FIFO small, or create some external lock. Unfortunately, both of these solutions eventually require `jeeves`, to be setuid, or for the butler to have publically writeable control areas.

To keep amount of data written to the FIFO small, an external file, containing the command and other control information is created, and the butler is advised of the file request via the FIFO. System V `lp` spooling uses a similar mechanism.

An "advisory" lock could be created, indicating that the FIFO is in use. If the lock exists, then other writers wait for it to disappear. The most "secure" lock which can be easily manipulated via the shell is a directory. Even `root` cannot `mkdir` if it already exists. Such a directory usually contains a file holding the process id of the locker, so that other processes can find out if the locker is still alive, adopting the lock if it isn't.

Such a scheme is in fact used by the butler to ensure that there is only one reader for each FIFO. It is still remotely possible for the lock not to work. Creating a directory, where the locker's process id is part of the name is probably a better solution as the lock and id would be created as a single atomic system event.

9. CONCLUSIONS

The butler is a typical software project. It's never finished. There are numerous desirable extensions and enhancements, especially in the area of secure privileged service agents.

Implementation of the butler has been mainly via shell scripts, illustrating the power of the shell as a programming and development tool. Where certain functions could not be effectively implemented by the shell, a minimal C program was written to take over only that function, in keeping with the UNIX paradigm.

Sources are shown in full in APPENDIX A.

10. ACKNOWLEDGEMENTS

Thanks to all the people at MetaPro Systems for their patience during debugging, to AUUG for providing a forum to present this material, and to Chris McDonald (chris@budgie.cs.uwa.oz.au) for doing such a great job on making this paper look so good.

11. APPENDIX A - SOURCES

wtail.c

```
/*
 *   wtail - waits on the standard input to grow
 *   and puts it on stdout.
 */ main () {
    char buf[1];

    while (1) {
        while ( read(0,buf,1) <= 0 ) sleep(1);
        if ( write(1,buf,1) == -1 ) exit(1);
    }
}
```

jeeves,

```
: #####
#   Copyright (c) 1989,90 #   All Rights Reserved by #   Metapro
Systems,           Perth,           Western           Australia. #
#####
#   Send a message to the butler

name=`basename $0` name=${name:-jeeves,}

# get defaults from the environment  BSPOOL=${BSPOOL:-/usr/spool/butler}
BFIFO=${BFIFO:-`expr $name : '),'`} PAUSE=3

# set standard error message usage="usage: $name <instructions>"

if [ ! -d $BSPOOL ] ; then
    echo "${usage}10pool directory $BSPOOL does not exist" 1>&2
    exit 3 fi if [ ! -x $BSPOOL ] ; then
    echo "${usage}10pool directory $BSPOOL is not searchable" 1>&2
    exit 3 fi

if [ -z "$*" ] ; then
    echo "Yes? request='line'"
    [ -z "$request" ] && exit 0 else
    request="$*" fi

cd $BSPOOL # Hurray!

# got a pipe? if [ ! -p $BFIFO ] ; then
    echo "${usage}46ipe $BFIFO does not exist" 1>&2
    exit 4 fi if [ ! -w $BFIFO ] ; then
    echo "${usage}46ipe $BFIFO is not writeable" 1>&2
    exit 4 fi

trap 'echo "no butler in residence"; exit 1' 15

parent=$(( export parent exec 2>/dev/null (sleep $PAUSE ; kill -15 $parent) &
echo "$request" >$BFIFO kill $!

# surprise! the butler speaks [ `date +%S` -eq 23 ] && echo "very good"

exit 0
```


butler

```
: ##### #
# Copyright (c) 1989,90 # All Rights Reserved by # Metapro
Systems, Perth, Western Australia. #
##### #
# It services requests read from a fifo in the specified # spool
directory.

PATH=/bin:/usr/bin:/usr/local/sbin:/usr/local/bin export PATH

# get defaults from the environment name=`basename $0` # startup command is
called "butler" default name=${name:-butler}

BSPPOOL=${BSPPOOL:-/usr/spool/$name} # spool dir for this butler BFIFO=${BFIFO:-
jeeves} # the named pipe BLIB=${BLIB:-lib} # directory for extended
commands NOTIFY=${NOTIFY:-root} # user to notify when in trouble

# set standard error message usage="usage: $name [-n name] [-s spooldir] [-f
fifo_name]"

if [ -t 0 ] ; then
    echo "$usageOutler must not be run from a terminal"
    exit 1 fi

# redirect input and ourput to meaningful places exec 1> /dev/console exec 2>
/dev/console

# process command line options set -- `getopt n:s:f: $*`

if [ $? != 0 ] ; then
    echo $usage 1>&2
    . but.exit 1 fi

for i in $* ; do
    case $i in
        -n) NOTIFY=$2 ; shift 2 ;;
        -s) BSPPOOL=$2 ; shift 2 ;;
        -f) BFIFO=$2 ; shift 2 ;;
        --) shift ; break ;;
    esac done

# check that the universe exists as promised

user=`grep ${NOTIFY}: /etc/passwd | wc -l` if [ $user -ne 1 ] ; then
    echo "${usage}otify requires local user name" 1>&2
    . but.exit 2 fi

if [ ! -d $BSPPOOL ] ; then
    echo "${usage}l0pool directory $BSPPOOL does not exist" 1>&2
    . but.exit 3 fi if [ ! -x $BSPPOOL ] ; then
    echo "${usage}l0pool directory $BSPPOOL is not searchable" 1>&2
    . but.exit 3 fi if [ ! -w $BSPPOOL ] ; then
    echo "${usage}l0pool directory $BSPPOOL is not writeable" 1>&2
    . but.exit 3 fi

cd $BSPPOOL # Hurray!

# got a pipe? if [ ! -p $BFIFO ] ; then
```

```

echo "${usage}48ipe $BFIFO does not exist" 1>&2
. but.exit 4 fi if [ ! -r $BFIFO ] ; then
echo "${usage}48ipe $BFIFO is not readable" 1>&2
. but.exit 4 fi

export BSPOOL BFIFO BLIB NOTIFY

# Check for an existing butler, or for a "concurrent" butler startup.

if mkdir lock 2>/dev/null >/dev/null ; then
echo $$ > lock/parent else
# a **hack**, but enough time for another lock to be created
# in the lock dir, in case of overlapping starts.
sleep 3
if butler.on 2>&1 >/dev/null; then
echo "${usage}907verstaffed: Another butler is serving" 1>&2
. but.exit 5
fi
if [ -f lock/parent ] ; then
pid=`cat lock/parent`
if kill -0 $pid 2>/dev/null ; then
echo "${usage}0nother butler is starting" 1>&2
. but.exit 6
fi
else
echo $$ > lock/parent # adopt lock
fi fi

echo "$name: EMPLOYED `date` at $BSPOOL/$BFIFO"

# read commands from the pipe and process them in subshells. wtail < $BFIFO |
while read cmd args ; do
command=`echo $cmd | tr "[A-Z]" "[a-z]"`
case $command in
the_butler_did_it) break ;;
silent) : ;;
hello) echo "$name: HELLO `date` at $BSPOOL/$BFIFO0command $args" ;;
*) ( but.servant $command $args)& ;;
esac
trap 'trap 15' 15 # wait for zombies
(sleep 1 ; kill -15 $$) 2>/dev/null &
wait
kill $! 2>/dev/null done

# finishing off:

echo "$name: FIRED `date` at $BSPOOL/$BFIFO"

while butler.on ; do sleep 3 ; done 2>&1 1>/dev/null

sleep 5 rm -rf lock # get rid of my stuff exit 0 # normal exit - honest!

```

but.servant

```
:
##### #
#   Copyright (c) 1989,90 #   All Rights Reserved by #   Metapro
Systems,      Perth,      Western      Australia.      #
##### #
#   Called by butler to support commands which #   are not butler
built-in.

# get defaults from the environment BSPOOL=${BSPOOL:-/usr/spool/butler}
BFIFO=${BFIFO:-jeeves} BLIB=${BLIB:-lib} exec < /dev/null # disconnect
pipe

# set standard error message usage="usage: $0 <command> [<arguments>]"

case $# in
  0) echo "$usage" 1>&2 ; exit 1 ;; esac

command=`basename $1` # only execute commands in the BLIB directory
shift args="$*"

if [ ! -d $BLIB ] ; then
  echo "$usage|executeables directory $BLIB missing" 1>&2
  exit 2 fi if [ ! -x $BLIB ] ; then
  echo "$usage|executeables directory not searchable" 1>&2
  exit 2 fi

cd $BLIB

PATH=`pwd`: $PATH export PATH

# check for an executeable command

if [ ! -f $command ] ; then
  echo "$usage|ommand $command not found" 1>&2
  exit 3 fi if [ ! -x $command ] ; then
  echo "$usage|ommand $command not executeable" 1>&2
  exit 3 fi exec ./ $command $args exit 1
```

but.exit

```
:
##### #
# Copyright (c) 1989,90 # All Rights Reserved by # Metapro
Systems, Perth, Western Australia. #
##### #
# Waits to be killed and returns with nominal exit code in
# interrupt, after closing open file descriptors.

trap 'exit $*' 1 2 3 14 15

exec <&- >&- # close standard input & standard ouput

while : ; do sleep 3600 ; done
```

butler.on

```
:
##### #
# Copyright (c) 1989,90 # All Rights Reserved by # Metapro
Systems, Perth, Western Australia. #
##### #
# This checks to see if we have a butler working # The only
argument required, if not set in the # environment, is the name of
the FIFO.

# get defaults from the environment BFIFO=${BFIFO:-jeeves}

# set standard error message usage="usage: $0 [fifo_name]"

case $# in
  0) ;;
  1) BFIFO=$1 ;;
  *) echo "$usage" 1>&2 ; exit 1 ;; esac

exec ${BFIFO}, silent >/dev/null
```

An Update on UNIX-Related Standards Activities

Stephen R. Walli

Report Editor, USENIX Standards Watchdog Committee

April in Chicago

The April IEEE POSIX working group meeting was significant. The newly formed Project Management Committee enjoyed its first full working meeting. A new steering committee was formed to manage the thornier issues surrounding POSIX profiles. The long awaited first draft of a Language Independent Specification of IEEE 1003.1-1990 was delivered for review and comment by interested working group members. And of course the week was dominated with Sun Microsystems's and UNIX Systems Labs' (USL) Open Look project request being put up against the Open Software Foundation OSF/Motif project request.

Project Management Committee

Chicago was the first working meeting of the newly formed Project Management Committee (PMC). The PMC monitors existing TCOS-SS projects and reviews new PARS (Project Authorization Requests). They use a mentoring process, whereby a member of the committee is assigned to each new PAR and each current working group. Each PMC member has several to track, because of the current number of projects.

Once a mentor is assigned a new PAR, they aid the PAR presenter in making sure it is properly formatted, and that all supporting documentation is present and complete. The point is to ensure that no PAR fails to be accepted by the TCOS-SS Sponsor Executive Committee (SEC) for documentation problems.

If the PAR is accepted, the mentor continues to monitor the project to ensure that it is on track. A project that loses focus on the current scope would receive help to bring it back in line with its stated purpose. The PMC has no direct authority to mandate anything, however they can recommend that the SEC take certain actions.

Members of the PMC include: Jason Zions, Shane McCarron, Lorraine Kevra, Roger Martin,

Derek Kaufman, Robert Bismuth, Don Cragun, and Tim Baker.

The PMC covered a lot of ground in its first week, starting on Sunday afternoon. The competing project authorization requests (PARs) to create standards for the two major X interfaces were discussed. (Discussion of the competing PARs follows.)

The POSIX.2 (Shell and Utilities) working group had a new PAR proposed, POSIX.2b, which covered the reformatting of the POSIX.2 and POSIX.2a (User Portability Extension) documents, and the incorporation of new utilities. The new POSIX.2b PAR was changed so that only new extensions would be part of the PAR, and the document reformatting was left out. This means we won't have a two thousand page document arriving for ballot as POSIX.2b! POSIX.7 (System Administration) was reviewed and recommendations made to separate it into several PARS under the same working group. An additional PAR for 1224 to cover the Object Management API for X.400 and X.500 was recommended. The POSIX.4, POSIX.6, and POSIX.11 projects were also reviewed during the first week.

This committee will likely begin to have more and more effect on the building of POSIX as it becomes comfortable with its role. Its members are long-time POSIX people with considerable experience and I look forward to what they bring to the overall process with all of its current problems of coordination and synchronization.

PAR Wars

Competing X Window PARS dominated the Chicago meeting. A month before the Chicago meeting, the Open Software Foundation (OSF) submitted a PAR to the TCOS-SS SEC proposing a direct ballot of the OSF/Motif API Document and Style Guide.

These documents would be reformatted according to TCOS style guides if the PAR was accepted. Test assertions and language independent

specifications would be written at OSF's expense if required. The legal copyright issues were arranged with the IEEE Standards Office. Sufficient industry acceptance and experience to make Motif a standard was claimed.

This forced the backers of Open Look to rush forward with a similar PAR, championed by Sun Microsystems and UNIX Systems Labs. Similar claims of industry acceptance and experience were made, and similar reformatting, test assertions, and LIS were promised. So the battle was joined once again.

There is significance to a direct ballot. POSIX standards are usually drafted by a working group who take base documents and create a new revised document. This revised document is balloted, reviewed, and made into a standard. With a direct ballot, there is no working group formed to build a document through a consensus process, but a balloting group is formed directly. In theory, the document is ready to be shipped to balloters, which was not the case for either of these PARS. TCOS-SS has rules for creating standards this way, but no one has ever done it before. The stage was set for a week of theatrics.

The first group to have to deal with the duel was the PMC. They stuck to the letter of their mandate, and reviewed these PARS to ensure they were correctly formatted. They also recommended that certain steering committees review them. The Steering Committee on Windowing User Interfaces (SCWUI) was an obvious reviewer. (*Yes, it's pronounced "scwewy", you wascally wabbit.*) SCWUI stated that it did not want these PARS accepted because of the overlap with the current P1201.1 (Windowing Toolkit API) work.

The Steering Committee on Conformance Testing (SCCT) was also asked for comment, and reported they had no concerns with either of them as stated.

One SC that was missed was the Distributed Services Steering Committee (DSSC) which came to light in the SEC discussions of the PARS. The Sun/USL PAR characterizes Open Look as a *distributed* desktop paradigm, so DSSC should have an opportunity to comment on it.

The P1201.2 working group is building a Recommended Practice document for driving window-based applications. The chair of this

working group raised concerns that if either of these documents became a standard before P1201.2 completes its work, there may be some conflict.

People discussed and debated all week in the hallways as to what would and should happen. Many felt that both PARS should be accepted, pointing to the IEEE 802 LAN standards as an example. Fortunately, many of the Europeans present were able to point out the problems with this, since they are currently living in a situation where many conforming implementations of OSI protocols cannot talk to one another because of such differences. This destroys any hope of building very portable applications which have windowed interfaces, unless one is willing to only be portable within windowing environment "A."

Others felt that neither PAR should be accepted, pointing out that if P1201.1 (Windowing Toolkit API) has been deadlocked over this type of API for so long, perhaps there isn't sufficient industry consensus to create a standard. On a few occasions during the week I heard different people refer to the original POSIX work to build POSIX.1. These references came about during completely separate discussions on conformance for language independent specifications and profiles. People talked about the way that the working group members laid aside their preferences for their particular flavours of UNIX in favour of building the standard. This does not appear to be happening in this arena.

Neither PAR could be accepted alone, since this would have disastrous commercial effects on the "loser." This points out some of the problems of allowing vendors and vendor consortia to produce such documents for standardization. It has been successfully done in the past in other areas of technology, but it needs to be done with great care, and not in an environment of direct and blatant commercial competition.

The membership of the balloting groups for these PARS would be interesting to see. The IEEE has rules about the percentage of balloting group content that is vendor related, user related or "general interest." This has never been contested in the past. Likewise, ballot resolution of comments and objections would also be interesting, as the PAR presenters would be responsible for administering their own ballot resolution according

to the PAR's scope. Somewhat like handing a pit bull terrier its own leash and telling it to walk itself without getting into a fight.

The SEC was forced into a painful discussion for a few hours on these PARs. During part of the discussion, PAR presenters pointed out that if the TCOS-SS SEC refused the issue, there was still a court of final appeal, being the IEEE Standards Board itself.

Fortunately, Paul Borrill was present. Paul is the vice-president for standards in the IEEE Computer Society, and a member of the IEEE Standards Board. Paul didn't have a lot to say, but his points were clearly made. First, he encouraged the groups to fix their own problems. Second, he reminded them who sets the rules, if people chose to bend or abuse them. (The IEEE Standards Board!) Points taken.

In the end, the discussion was halted with a flurry of Robert's Rules procedural magic. The Rules are used as a way to ensure that work is accomplished in a committee forum and that all participants have fair opportunity to be heard. The SEC resolved that it "does not feel at this time that it should sponsor either the Modular Toolkit Environment PAR (*Motif*) or the Open Toolkit Environment PAR (*Open Look*). The PARs are in procedural limbo. By that hour, the SEC was only too happy to kill discussion of the PARs. The PARs have not been tabled, withdrawn, or postponed. They are still very real and I fear that the Santa Clara meeting will have these PAR presenters haunting the hallways, singing "wee're baaaack...."

Profiles! Get Your Profiles!

For quite some time now, profiles have been a great source of confusion in the POSIX world. Ask ten different people from ten different areas what a POSIX profile is, and you will indeed receive ten different answers. There is a list of serious outstanding issues on defining, coordinating, and standardizing POSIX profiles, which has been built up by the working group on the POSIX Guide (P1003.0) and current profile writing groups.

They have long felt that some form of managing group needed to take charge of these issues. After much (circular) discussion as to the nature

of this committee (is it a rapporteur group, an ad hoc group, or a steering committee?) it was finally decided that a steering committee was required to deal with the management issues of profiles. The SEC ratified this decision and the Profiles Steering Committee was born.

Bob Gambrel is the chair of the Profiles Steering Committee, and each working group with a profile project also has representation. The group held its first organizational meeting the next day and by the time Santa Clara rolls around, the committee's work will be well under way.

LIS POSIX.1

A first draft Language Independent Specification of POSIX.1 (System Application Program Interface) was delivered this week. Language independence is an issue raised by ISO who wish standards to be unrelated to a particular programming language.

In January, the SEC formed a subcommittee to solicit and evaluate submissions to produce a complete POSIX.1 language independent specification (LIS). Monies were put forward by the IEEE Computer Society, the contract was awarded, and the work was done.

The completed first draft language independent specification of POSIX.1 (to IEEE 1003.1-1990) and a near complete draft C language binding (POSIX.16) were presented at the LIS BOF on Monday afternoon. BOF attendees raised concerns that input on certain language independence issues raised over the last few working group meetings may not be completely reflected in the drafts, but the drafts were generally well received. Copies were in such high demand that the rules for making document copies at meetings had to be changed to prevent further drafts from being produced.

A concrete example of a near complete LIS of POSIX.1 now exists. Other working groups can use it as an example in much the same way that POSIX.3.1 (Test Methods for POSIX.1) is an example of how to construct and structure test assertions. Many working groups point to the functionality described in POSIX.1, and it was unclear how their LIS would need to be structured to point to an LIS version of POSIX.1. These issues can now be addressed and the language independence re-

;login: 16:4

quirements on the POSIX standards process can move forward with more confidence.

ISO's working group 15 (WG15) on POSIX requested that language bindings to the POSIX standards come forward as "thin" bindings to the LIS documents. Thin bindings indicate that there is no significant duplication of semantic content between the LIS and the language binding. Because of this request, the POSIX.5 (Ada Binding) and POSIX.9 (FORTRAN-77 Binding) working groups are not proceeding at the international level at this time.

Both of these groups are balloting their documents at the IEEE level and are busy resolving ballot objections. Both of these groups have language standards groups reviewing their respective programming languages, with a view to significantly changing them. The groups feel they can better serve the industry by waiting until the POSIX.1 LIS and the changing language standards stabilize, and then produce the documents which will be forwarded to the international level for standardization. In the meantime, IEEE standard bindings will exist for Ada and FORTRAN-77 to the C-based POSIX.1 standard.

Report on 1003.0: POSIX Guide

Kevin Lewis <klewis@gucci.enet.dec.com> reports on the April 15-19, 1991 meeting in Chicago, IL:

Summary

POSIX.0, more familiarly referred to as 'the Guide' is best summed up by the first sentence of Draft 11. "This guide identifies parameters for an open system environment using the POSIX operating system/application interface as the platform.

The working group spent the week reviewing the document, addressing omissions and readability issues. Careful attention was paid to the guide's readiness for mock ballot for eventual submission to ISO in October, as a technical report.

Report

Believe it or not, this group made its best forward progress by reviewing the guide docu-

ment backwards. I'm still trying to figure out what this says about our group. [*ed - And so are we all!*] This forced us to deal with issues that were latent because we simply had not made it all the way to the end of the document before. On the occasions we did, we were too exhausted to do anything substantive. There were times during the review when I felt we were writing a very succinct and precise "ballad." Other times we seemed to be writing the sequel to "War and Peace." Overall we made significant progress. Many key issues were addressed in Chicago.

First was the errant and unintentional (I think) omission of the balloting P1003.2 (Shell and Utilities) standard from the guide. Wendy Rauch agreed to draft a write-up on how this standard fits into the context of the guide for its next release.

Another issue was that of how to address character-based terminals in the user interface section. Pertinent contributions are being written for inclusion in the next draft.

The use of the guide as an ISO Technical Report was also discussed this week. Factors affecting this are the guide's readiness and whether or not this readiness coincides with an acceptable time frame for ISO. There is a document synchronization plan between the IEEE and ISO, which will allow POSIX documents to be published concurrently as both ISO and IEEE standards. POSIX.0 plans to use a mock ballot as a way to judge its readiness. The group agreed that this ballot could not commence before the October '91 meeting. The group may, however, submit the guide to ISO prior to the completion of the mock ballot.

As you might imagine, the decision to submit the guide to ISO is very subjective and discussion of this will probably eat up considerable time at the October meeting. (This reminds me. I better get Mr. Isaak to provide me with a large gavel.)

Lastly, POSIX.0 strongly focused its attention on the overall readability of the guide in such a manner that I felt we were finally able to see the proverbial "forest for the trees." This will be the primary focus in the July meeting, strongly coupled with a review of those sections that should be either dropped (e.g. the graphics section) or postponed (e.g. the languages section) until after the mock ballot. (The languages section is likely

to be postponed due to lack of help and not because it is any less significant.)

In summary, POSIX.0 is on track, heading in the right direction, BUT with some medium-to-high hurdles remaining.

Report on IEEE 1003.2: Shell and Utilities

David Rowley <david@mks.com> reports on the April 15–19 meeting in Chicago, IL:

Background

A brief POSIX.2 project description:

POSIX.2 is the base standard and deals 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.”

POSIX.2a, the User Portability Extension or UPE, is a supplement to the base POSIX.2 standard and standardizes commands, such as *vi*, that might not appear in shell scripts but are important enough that users must learn them on any real system. It is essentially an interactive standard, and it will eventually be an optional chapter to a future draft of the base document. This approach allows the adoption of the UPE to trail Dot 2 Classic without delaying it.

Some utilities have both interactive and non-interactive features. In such cases, the UPE defines extensions from the base POSIX.2 utility. Features used both interactively and in scripts tend to be defined in the base standard.

POSIX.2b is a newly approved project which will cover extensions and new requests from other groups, such as utilities for the POSIX.4 (Realtime) and POSIX.6 (Security) documents.

Together, Dot 2 Classic and the UPE will make up the International Standards Organization’s ISO 9945–2—the second volume of the proposed ISO three-volume POSIX standard.

Summary

POSIX.2 (Shell and Utilities) closed its recirculation ballot on March 29. The Chair feels there will only be a small number of changes to the current draft, probably circulated as change pages. There were some ballot concerns over the internationalization areas, but these will likely remain intact due to current support. There was also a concern raised over the ballot period for a 900+ page document. POSIX.2A closed its recirculation ballot on April 24.

POSIX.2b has been approved after a number of scope change recommendations from the PMC. The POSIX.2 group requests technology for both a new archive format, and also a new family of compression utilities. Much of the Chicago meeting was spent with POSIX.3.2 (Test Methods for POSIX.2) creating test assertions for the document.

Status of POSIX.2 Balloting

The Draft 11 Recirculation Ballot for POSIX.2 closed March 29. Some balloters seemed to forget that it was a recirculation ballot, as there were a large number of objections on items other than changes. These were ruled unresponsive.

Hal Jespersen, the POSIX.2 Chair and Technical Editor, believes that there will only be a small number of actual modifications to the draft. Draft 12 (which may possibly be called Draft 11.1) will likely be distributed as a set of changed pages, which he estimates to number about 200. (More recent estimates suggest the number of pages to be as low as 50.) Expect it sometime around July.

There were a number of objections to the internationalization part of POSIX.2, but since Hal has support from X/Open, WG15, etc., he thinks the current specification will remain largely intact.

There was a problem with the Draft 11 distribution. POSIX.2 is now over 900 pages. Its balloting period was 30 days, although with a mailing lead it was about 6 weeks. Due to postal services, some members of the balloting group received their ballot copies only two weeks prior to the closing deadline. Hal Jespersen was as accommodating as he could be on the deadline, but the extent of these submissions was definitely affected. The question rears its head again. Should

;login: 16:4

we be balloting POSIX standards the same as we ballot smaller hardware standards?

The ISO standards process sees a document move through three phases on its way to standardization — Committee Document, Draft International Standard, and finally International Standard. Draft 9 of POSIX.2 is currently being used as a committee document. The ISO has requested the U.S. Member Body to forward to them another draft once it has become more stable. The next draft (Draft 12 or Draft 11.1) will be a likely candidate. The ISO has delegated responsibility for producing the POSIX draft standards documents to the U.S. Member Body, ANSI, which in turn delegated the responsibility to the IEEE.

Status of POSIX.2a Balloting

The Draft 6 Recirculation Ballot of POSIX.2a (UPE) closed April 24. Unfortunately the deadline for comments came a mere three days after the end of the April meeting. There were quite a few comments on the unfortunate timing of the ballot close. Work on ballot resolution is ongoing.

New PARs

The Project Management Committee (PMC) has recommended that the proposed PAR (Project Authorization Request) for 1003.2b be split into two parts. One part will cover extensions and new requests from other groups, such as the `tar`, `cpio`, and new `pax` file formats from POSIX.1 (Kernel) and utilities from POSIX.4 (Realtime) and POSIX.6 (Security). The timing and alignment issues with the ISO 9945-2 balloting process will be covered by the other part.

The scope of this second PAR is restricted to standardization of existing industry practice. The group does not want to get into designing new utilities.

There is also concern over draft stability when discussing the commands to access the features from the POSIX.4 and POSIX.6 standards. How mature does the feature have to be before the POSIX.2 group goes to the effort of defining a command interface to it?

Discussion

Donn Terry, the chair of POSIX.1, officially handed off responsibility of the `pax` file formats,

including the new format currently being designed, to the POSIX.2 group.

A proposal was examined to reserve the utility status return code 126 to indicate that a utility was found but could not be successfully invoked. This would be especially useful in systems with limited resources, where execution can not be assured even though the utility has been found. The group generally agreed that this was reasonable.

There was a question as to whether the warning message for `getopts` should be specified in the standard or not, so that filters could parse it. It was decided to **not** specify the error format, since there is no precedent for stating the format of something written to standard error.

There was discussion on removing `-e` from `pax`, since charmaps were not really intended to be used in this manner. The `-e` option was intended to allow filenames to be written out using only characters from the portable character set. OSF had already implemented this in their `pax`, and agreed that it didn't work out too well.

The `$(` notation in the Korn Shell currently can have two widely different meanings: either spawning a subshell or expressing an arithmetic operation. The working group agreed that disambiguating by placing a space between the two parentheses, though inelegant, was the best approach.

There was some discussion on a proposal on User Controllable Limits, and whether or not it had relevance to POSIX.2. The group felt that there should be a command interface to these services, with the functional interface to be provided by POSIX.1. A proposal for the POSIX.2 interface is now being solicited.

We also discussed the `test` command. David Korn proposed fixing the functionality of `test` based on the number of arguments given (1, 2 or 3). Invocations with greater than 3 arguments would not be portable. We generally agreed on this approach.

Richard Hart from POSIX.7 (System Administration) presented the syntax for a new printing command based on the MIT/Athena Palladium network printing services. Everyone in the

POSIX.2 group agreed that the proposed syntax was awkward:

```
prpr -print-quality draft
```

means use draft if you can

```
prpr print-quality draft
```

means you *must* use draft

```
prpr p-q draft
```

means the same thing, but “print-quality” has been abbreviated.

The abbreviation mechanism is particularly poor, since it is likely that new extensions will cause namespace conflicts.

Requests for technology

There is an opportunity now to propose a new archive format. The only prerequisites are that it is ISO 1001 tape format compatible, and uses the ISO 646 character set. This consists of the invariant codeset from a variety of character set standards, largely 7-Bit ASCII minus about 10 contentious characters. Here’s a list of requirements:

- Should be textual (mailable) if members are all textual
- Should support filename and file contents mapping (eg. for dynamic encryption or compression)
- Should be extensible

Personally, I don’t understand why the ISO 1001 tape format needs to be a restriction. Archive formats have many other uses besides tape backup and transfer. To embed the tape format in what could otherwise be a general-use archive format seems overly complex and restrictive.

The group is also looking for a new family of compression utilities, now that the Lempel-Ziv-Welch family of commands have been removed from the standard. The main requirements for a substitute are:

- The algorithm should be expressed (expressible) in a language independent form
- The algorithm should be free of patent issues

Test Plans and Assertions

A test plan for POSIX.2 and POSIX.2a has been written, and has been passed to POSIX.3.2 (Test Assertions for POSIX.2) for comment and approval.

Tuesday to Thursday were spent writing test assertions in a joint meeting between POSIX.2 and POSIX.3.2. Commands tackled included `make`, regular expressions, `ln`, `cp`, `rm`, `mv`, `pax`, `pathconf`, `echo`, and `read`.

Some members volunteered test assertions written by their companies, usually to a previous draft. They were almost always unusable, either because they were out of date (based on previous drafts), or of poor quality. Writing good test assertions is very difficult, and quickly points out (the many) ambiguous wordings in the draft.

The POSIX.3.2 group would like to go to a mock ballot after the October meeting in Parsippany, New Jersey.

Report on 1003.3: POSIX Test Methods and Conformance

Andrew Twigger <att@root.co.uk> reports on the April 15–19, 1991 meeting in Chicago, IL:

Summary

The POSIX.2 (Shell and Utilities) working group made good progress writing test assertions this week, with POSIX.3’s (Test Methods and Conformance) help. Many working groups, however, are discovering that writing test assertions requires a non-trivial effort. This week also saw the delivery of the newly published “IEEE 1003.3–1991 - Test Methods for Measuring Conformance to POSIX.” Concerns are still being raised over NIST’s certification policies.

Report

Chicago will probably go down in history as the meeting where test methods invaded the POSIX working groups with a vengeance. After years of mild abuse and jesting (mostly aimed at NIST), the SCCT (Steering Committee on Conformance Testing) seems to be succeeding in the goal of ensuring that POSIX standards are balloted with test method specifications. Despite rumours during the week that a wake had been arranged for the SCCT Chair,

:login: 16:4

most of the screams were heard from working groups, who having been previously informed that test methods would be easy to write and would only take a couple of meetings, were finding that this was a far from straightforward task.

While most of the remaining members of the original POSIX.3 working group continued work with the remaining members of POSIX.2 in generating assertions for the POSIX.2 standard, a few of the POSIX.3 elders started helping other working groups to develop test methods for their standards. The POSIX.3.2 group (i.e. POSIX.3 + POSIX.2) met for three days during the week and spent all of that time writing assertions in small groups of three or four people.

Some of the more difficult aspects of POSIX.2 were tackled, specifically Basic Regular Expressions and the Make utility. Most of the smaller utilities have assertions written already although most of these need to be updated to align with the current draft. It is hoped that enough of the work will have been completed after the October '91 meeting to start internal ballot of the draft document with IEEE balloting commencing in the first half of 1992.

Other working groups that have started producing test methods include POSIX.4, POSIX.6, POSIX.8, POSIX.15, POSIX.17, and P1224.1, P1224.2. Most of these groups are at an early stage in their test method development and are producing a wide variety of problems for the "experts" to address. Several of these groups have noted that the formal process of producing test assertions has uncovered a variety of deficiencies in their drafts; so perhaps there is some benefit in test methods after all!

The highlight of the week was the arrival of the latest of the series of POSIX standards, IEEE 1003.3-1991. This document was made available at the extraordinarily discounted price of \$15.00 per copy, which works out to 30 cents a page! Still I suppose that considering the number of committee hours that went into the document, it's a real bargain. (One working group member calculated an industry cost in excess of \$5,000 per page.)

Other concerns which arose during the week relate to NIST's adopted certification policies and procedures. Many working groups continue to be

concerned about these. This has been a long running battle involving both accredited testing centres and implementation suppliers in assisting NIST in the refining of their policies.

The current major cause for concern is whether there would be equality in the certification process or whether a particular implementor would gain advantage from receiving the first conformance certificate. NIST was not explicit as to the procedures that they would employ to deal with the initial surge of certification requests, but made assurances that everybody would be satisfied when the process was completed. This seemed to satisfy nobody! We'll have to wait until Santa Clara to see whether NIST is really here to help us.

Report on POSIX.4, .4a, .4b, .13: POSIX Realtime Extensions

Bill O. Gallmeister <uunet!lynx!bog> reports on the April 15-19, 1991 meeting in Chicago, IL:

Summary

This week, the working group advised the technical reviewer for IPC Message Passing to either delete or severely prune back the IPC chapter. The large group also agreed to work closely with the POSIX.12 sockets group on their interface to ensure that a "Real-Time Protocol" could be implemented on top of sockets to meet real-time message passing requirements.

Work was done to harmonize POSIX.4 binary semaphores and POSIX.4a (Threads) mutexes and condition variables. A mutex is a lock semaphore, so that only one person has access to a resource at a time - MUTually EXclusive access.

We also began to explore work for POSIX.4b (the Yet More Real-Time proposal). Work here possibly includes the Ada Catalogue of Interface Features and Options (CIFO).

Work continued on the Application Profiles, Test Assertions, and the Language Independent Specification.

There will probably be a new recirculation of POSIX.4 before the Santa Clara meeting. POSIX.4a will probably not be recirculated before then.

Report

IPC

The IPC chapter in POSIX.4 is a bone of contention. In my estimation, it retains the largest number of unresolved negative ballots in all of POSIX.4. Most objections center on the fact that the interface doesn't look much like anything seen in UNIX before, and on doubts that the interface can be implemented efficiently.

A small group spent this week looking at IPC and ways to deal with it. They came up with some startling recommendations. First, they noted that the sockets interface, which most of us are familiar with from BSD, is currently undergoing standardization by POSIX.12 (Protocol Independent Interfaces). They noted that all the needs of real-time and transaction-processing IPC could be met by a new sockets protocol, perhaps with a few extensions to the sockets interface itself. There are generally two socket protocols on a UNIX system: the UNIX domain protocol, which communicates with other processes on the same machine, and the Internet protocol, which does the network thing. A real-time protocol would be akin to these. The small group recommended that we work with POSIX.12 to ensure that such a real-time protocol could be defined.

In addition, they made specific suggestions for trimming back the current IPC chapter, if it is not removed altogether. These suggestions included removing non-copy IPC modes and some of the more baroque asynchronous modes of the interface. Another option would be to delete the POSIX.4 IPC chapter entirely and await POSIX.12 sockets and a real-time extension on top of that—probably a three-year wait.

The votes, when taken, were 17–5 in favor of deleting the chapter, and 29–2 in favor of trimming the chapter severely. However, when given the choice of deleting POSIX.4 IPC or pruning it, the vote was 21 to 15 in favor of deleting, and only two working group members admitted that they would ballot against the draft if IPC was removed.

Synchronization

POSIX.4 specifies a binary semaphores interface; POSIX.4a (Threads) specifies mutexes and condition variables. These two facilities, while

rather similar in the abstract, are quite different in the current drafts. A group attempted this week to bring the two closer together.

Mutexes and condition variables are based in the memory of the process, while binary semaphores are accessed via an opaque object that might be a memory address, but might not. It had been noted in New Orleans that POSIX.4 binary semaphores worked between threads in a process, but that thread mutexes and condition variables did not work between separate processes. This lack of parity has been the source of many ballot objections to both POSIX.4 and POSIX.4A.

The small group came up with a model of how synchronization was expected to work in the vast majority of cases. Mutexes, condition variables, and binary semaphores are all implemented in user memory, much like how thread mutexes are currently implemented. In addition, an extension to this implementation allows the memory-based implementation to operate in shared memory between processes.

Because some machines (such as Crays) do not possess the hardware for memory sharing, a more abstract interface to process synchronization is required. (Those machines will not implement binary semaphores like most other people, but will do something different.)

The working group approved a number of small changes to harmonize POSIX.4 and POSIX.4a with regards to process and thread synchronization based on this model. The working group also demanded some documentation explaining the different models and requirements motivating the different facilities and interfaces. Hopefully, such documentation will clear up the confusion currently surrounding the two interfaces.

POSIX.4b

POSIX.4b has as its goal the standardization of some of the less mainstream features of real-time systems. These are basically areas that the POSIX.4 group decided to defer until "later." During this meeting, small groups worked on interfaces for timeouts on all blocking system calls, for enhanced memory allocation, and for direct application use of interrupts. The documents for all three of these areas are quite immature, and the small groups spent their time trying to identify

;login: 16:4

models and requirements. I believe the first draft of POSIX.4b will be generated in Santa Clara. Other possible work items for this proposal include extensions to the existing synchronization primitives, and the Ada Catalogue of Interface Features and Options (CIFO).

The timeouts group received some conflicting advice. Many people do not want this interface at all. Of those who did, there was strong consensus for new function calls for each blocking call, i.e., we'd have `timeoutread()`, which could time out after a certain interval of time, since `read()` is a blocking call.

The memory allocation group is concerned with being able to allocate from specific pools of memory—memory presumably having some special characteristic. They were directed to see whether `mmap()`, from the Shared Memory and Mapped Files chapter, would suit the requirements.

The interrupt access group came up with a model of something like signal handlers for attaching a process directly to an interrupt. Additional semantics of the interface still need to be defined, (e.g. can system calls be made from a user "interrupt handler").

Application Profiles

The real-time applications profiles group is well on its way to producing a draft which defines multiple profiles: an embedded profile, a profile one up from that, a mid-size profile, and a kitchen sink profile.

The kitchen sink profile is easy: it includes everything. At the lower layer is an embedded profile which will hopefully be very small. It specifies the threads interface, but would like to not include the process interface, i.e. no `fork` or `exec`. It has `read`, `write`, and `open`, but no other file interface. The target for such a system would be an embedded system, perhaps without an MMU. Much of POSIX.1, and in fact much of POSIX.4, is irrelevant to such a system. The largest area to be addressed now is the ability to remove pieces from POSIX.1 (i.e., `fork()` and `exec()`) and still have a "POSIX" system. POSIX.1 is not set up to allow such selective removal of interfaces.

Test Assertions and Language Independent Specifications

Small groups (of one each) continued to work on the test assertions and the language independent interfaces for POSIX.4. Not much progress was made, due to the pressing requirements of other issues and the fact that much of this work is best done late at night hunched over one's terminal. This work will continue and should be more advanced at the Santa Clara meeting.

Report on POSIX.6: Security Extensions

Ana Maria De Alvare <anamaria@sgi.COM> reports on the April 15–19, 1991 meeting in Chicago, IL:

Summary

The POSIX.6 group spent the week preparing draft 11 of their document for internal mock ballot. They began work on their test assertions document. The IEEE balloting group formation process is now officially closed.

The Privilege subgroup discussed a proposal to remove the global constant `POSIX_PRIV_EFFECTIVE` from the draft. The Audit subgroup will not be able to address the portable audit format before balloting begins, but they will define the audit trail header. The liaison group between POSIX.6, POSIX.7 (System Administration), and the Distributed Services groups will report back to the TCOS-SS Sponsor Executive Committee (SEC) at the July meeting, recommending that a new coordination group be formed.

Report

The POSIX.6 group met for the entire week in Chicago. The group concentrated their efforts on cleaning up draft 10 of the document. The balloting solicitation process has been closed. If you requested to be in the balloting group, please confirm you are on the list by calling the IEEE, Anna Kacznarek (908-562-3811).

A major action item was the creation of the test assertions document for POSIX.6. This will be a separate parallel document. The definitions and overview sections of POSIX.6 were addressed this week. Each subgroup will be responsible for creating the test assertions for the document sections

they are working on. The subgroups will maintain consistency between the test assertions and the POSIX.6 document. Modifications to the POSIX.6 document will signal modifications to the test assertion document.

In the next meeting we are planning to integrate test assertion sections from POSIX.3.1 (Test Assertions for POSIX.1) into our document. Dave Rogers (Data Logic) and I are co-chairing this effort. If you are interested in participating in the test assertion work, please let me know (anamaria@sgi.com or 415-335-7309).

POSIX.6 will mock ballot draft 11 within the working group before July. We plan to review written comments to this mock ballot at the July meeting. If all the written comments are addressed, we will try to ship the document for IEEE ballot after July. We could then start resolving the ballot objections at the October meeting.

Privileges

Secureware's VP of Marketing proposed eliminating from the standard the system global constant, `POSIX_PRIV_EFFECTIVE`, which turns on/off all the privileges already set by the process or set by the file privileges in effect. The system global constant can increase or decrease the effective privilege set.

The argument against the system global constant was that when `POSIX_PRIV_EFFECTIVE` is on, a privilege aware program (i.e. a trusted application) will have effective privileges on before it uses them. This violates the concept of least privilege, since the process contains more privileges than it needs. It is the responsibility of that trusted application to turn off all effective privileges and then turn them on one by one as it needs them.

Another argument against the global constant is that it gives the system manager a central point to turn on/off privileges. With the new scheme, programs that turn "priv_effective" on are consciously given permission to do so, a point that brings higher granularity.

A vote was taken and the group decided to eliminate the system global constant, `POSIX_PRIV_EFFECTIVE` and use "priv_effective" as an additional file privilege. The standard now contains three privilege sets associated with a process (in-

heritable, permitted, effective) and two privilege flags ("allowed" and "forced") associated with each privilege on a file. The two file privilege flags are:

—Allowed - a flag associated with a file privilege that will authorize it to be on during the execution of that program, if the process possesses that privilege.

—Forced - a flag associated with a file privilege that will be on during the execution of that program even if the process does not possess that privilege. This allows for old setuid programs to continue to work under POSIX.6 without source code modifications.

The new file privilege "priv_effective" will turn on the process's effective privilege set. If your file has "priv_effective," your file makes effective all of the privileges that are on after calculating "allowed" and "forced" flags against the process's inheritable flags.

A process possesses three sets of privilege flags: *inheritable*, *permitted*, and *effective*. For a process to access a file, the process's effective privilege set (built from its inherited and permitted sets) is tested against the file's privilege set. To be able to pass a privilege from the *inheritable* set (from its parent process) to the *permitted* set, the system will test the process's inheritable privilege against the file's "allowed" and "forced" flags for that privilege. If the file privilege's "allowed" flag is set, then the privilege is turned on in the process's *permitted* set. If the file privilege's "forced" flag is set, then the privilege is turned on in the process's *permitted* set even if the privilege was not inherited.

To be on in the process's *effective* set, the system compares the inheritable privilege against the file's "allowed" and "forced" flags. If the process's inherited privilege is in the file's "allowed" set and the file's "priv_effective" privilege is set, then the privilege becomes effective. If the process's inherited privilege is in the file's "forced" set and the file's "priv_effective" privilege is set, then the privilege becomes effective. In other words, to be set effective the file's "priv_effective" flag must be on.

Some of you might think that this scheme still gives me a trusted application with effective privileges turned on. The list of programs with

;login: 16:4

privileges turned on, however, is smaller than using the system global constant. In addition the effective privilege set is not on for all processes.

All of this can become very confusing. Sometimes I have trouble understanding all of the benefits. Every time I read the document new questions come to mind. Sometimes I agree and other times I don't. Hopefully the mock ballot will call attention to any ambiguous areas left in the draft document.

Access Controls

Both the discretionary and mandatory access control subgroups (DAC and MAC) are ready for our internal mock ballot. The primary DAC related changes for draft 10 concerned default access control list (ACL) behavior and the command `chac1` which changes the ACL. The MAC group had no hot issues to discuss.

Audit

The Audit group finished modifying the draft and writing the rationale for integrity protection, header flexibility, and cross references. The group felt they cannot address the portable audit format before balloting; however, they are planning to define the audit trail header containing:

- POSIX audit indicator field,
- version ID,
- data format indicator (type XDR, little endian, big endian),
- time zone offset,
- machine id, and
- audit style.

The audit file format remains up in the air.

POSIX.6/POSIX.7/Distributed Services Liaison

The liaison group met on Wednesday. Mike Ressler stepped down and I became the chair of the group. We discussed the status of the group and what we should bring forward to the TCOS-SS Sponsor Executive Committee (SEC). Everyone agreed that we have enough information to create a report to the SEC discussing the problems we discovered and to make recommendations.

I will present our report at the July meeting with the help of the liaison group. The report will

include an overview of each subgroup's objectives, a list of problem areas discovered during our meetings, and recommendations to solve these problems. I hope that SEC acts upon our recommendations.

One recommendation we want immediate action on is the lack of a mechanism to ensure that one POSIX extension can interoperate with another POSIX extension. An example of this interoperability issue is having POSIX.6 and POSIX.8 (Transparent File Access) on the same system. We are proposing a new group be formed which will check that POSIX standards interoperate with each other or to at least document where different POSIX extensions cannot interoperate.

1003.7: System Administration

Martin Kirk <m.kirk@xopen.co.uk> reports on the April 15–19, 1991 meeting in Chicago, IL:

Summary

POSIX.7 is getting back on its feet again, having come through a rocky period in its history. The Project Management Committee (PMC) has reviewed the project and recommended that it be split into a number of sub-projects, organized by POSIX.7. Likely candidates are print management, software management, and user environment management.

Report

The April 1991 POSIX meeting in Chicago may turn out to be the final step in the rehabilitation of the POSIX.7 Systems Administration working group.

Probably as a result of its occasionally controversial past, POSIX.7 was among the first batch of working groups to be reviewed by the newly created Project Management Committee (PMC).

It is possible to speculate on whether POSIX.7 would have met the PMC's project approval criteria had it been in existence two years ago. One of the most pertinent criteria would probably have been the existence of a suitable base document. A likely candidate would have been the NIST-proposed draft System Administration document, though it might have been difficult to demonstrate the right kind of consensus around it!

Anyway, the PMC was not in existence then and POSIX.7 was duly created. The first couple of meetings were spent investigating the possibility of standardising the existing systems administration commands that we all know and love. The working group decided that there was little benefit to be gained from solving the single machine problem in a world that was rapidly moving towards a norm of heterogeneous networks, and set off on its trek into the rather more esoteric realms of object-oriented systems management for networks of heterogeneous machines.

Inevitably this change of direction led to charges that the group was inventing hand-over-fist, rather than following the "traditional" standards model of codifying existing practice. (No-one ever argued that the group had gone beyond its scope, which was cunningly worded to allow the group to do almost anything.)

Moving into the world of distributed systems management opened up various cans of wriggling things with labels like "interoperability" and "frameworks." (This was when I discovered that rat holes were full of worms.) It was at this point that an over-enthusiastic embracing of object-oriented concepts led to the promulgation of a command line interface that was tremendously orthogonal, but completely different to all known existing practice.

Interoperability proved to be a particularly thorny problem. Everybody could agree that it was essential, but there was no emerging consensus as to how it would be achieved.

In hindsight, this was the lowest point of POSIX.7's fortunes. From this point the rehabilitation commenced. The first stage was an agreement among the group to limit the scope of its activities (but not its objectives). The group decided to concentrate on two particular aspects, the definition of the managed objects required for systems management, and the definition of management tasks — the administrator's view of the job in hand. This decision allowed the group to close the door on the rat holes and concentrate on areas where it was able to make progress.

Part of the motivation for this decision was recognition that the problem space is vast and that trying to attack it over too large a front was a suicidal maneuver. There was also an increased

awareness of the related work of other organizations, such as the OSI Network Management Forum, the OSI Implementer's Workshop Network Management SIG, and x/Open. As this other work comes to fruition, it will be available for use by POSIX.7 and will likely solve some of the thornier problems, such as interoperability.

So what happened in Chicago to raise hopes that the rehabilitation is almost complete? For some time the group had been aware that some functional areas were much closer to reaching a consensus than others, and it had been considering how it might better organize the work in order to "get something out of the door." The result of the PMC review of POSIX.7 was a recommendation that the existing project should be split into a series of sub-projects, each representing a functional area within the overall problem space, and each leading to a separately balloted document. The existing project would be retained as an "umbrella" to handle the coordination issues arising from the split. This is necessary if the parts are to form a coherent whole. New projects would be raised to cover a first set of functional areas. No more than two or three of these functional sub-projects would be active at any time. This would keep the group focussed on a set of limited and achievable goals. New projects would be instantiated as existing ones move into the balloting phase.

One of the benefits of this approach is that each of the new sub-projects must pass the PMC's project approval criteria before it is recommended. The proposal will be properly scrutinized to ensure that the project is likely to succeed within reasonable timeframes. A result of the earlier decision to concentrate on managed objects and management tasks will be to relate the new projects much more closely to existing interfaces, thus removing one of the rods which the group had fashioned for others to beat it with. An obvious source of candidate management tasks can be found in the existing administrative command set on the systems around us, and it would be a perverse decision indeed to introduce gratuitous changes to the style of that interface.

The first set of sub-projects are likely to be Print Management, Software Management, and User Environment Management. These three

;login: 16:4

represent areas where the work of the group is well advanced and where there is strong commitment of energies.

The Print Management work is based on the MIT Palladium printing system, which has the benefit of being well-aligned with the emerging ISO distributed printing standard, DIS 10164. The Print Management sub-group within POSIX.7 has been working with the Palladium documents for over a year and this work is probably the closest to being complete.

Software Management has enjoyed a resurgence of interest within POSIX.7 over the last 6 months, with source material being drawn from DEC, HP, AT&T, and Siemens-Nixdorf. The small group that has been working in this area has been comparing the various technologies and (not surprisingly?) finding a great deal of commonality between them in terms of their underlying concepts and functionality. The task of identifying a common model and a common set of functions is well advanced and bodes well for the future. (Indeed, the rate of progress is positively alarming!)

The third area, User Environment Management is a logical candidate for inclusion in the initial set of sub-projects. Much of systems management is concerned with the management of users and their interactions with other components of the system. Many management tasks need to be able to refer to users and it seems to be appropriate to tackle this area at an early stage. (For some inexplicable reason, the "add user" operation seems to be the universal example always brought up when talking about some aspect of systems administration - another motivating factor.)

Looking beyond the confines of POSIX.7 into the wider world, the original decision to adopt an object-oriented approach to the problem of systems administration is at last being vindicated. Object-oriented concepts lie at the heart of the OSF Distributed Management Environment request for technology (RFT), the UI Systems Management SIG, and the X/Open Systems Management working group. It looks as if history will show POSIX.7's decision to have been a far-sighted move rather than turning up a blind alley.

Report on 1003.9: POSIX Fortran-77 Bindings

E. Loren Buhle, Jr., Ph.D.

<buhle@xrt.upenn.edu> reports on the April 15-19, 1991 meeting in Chicago, IL:

POSIX.9 met to resolve objections and comments raised to the first ballot of the FORTRAN binding to ISO/IEC 9945-1 Standard (also known as POSIX.1). The ballot began in late December 1990 and ended on February 20, 1991. This first proposal did not obtain the necessary 75% acceptance of the balloters. There were 73 people in the total balloting group, of which 56 were eligible to vote on the standard. The others were parties of interest. Of the official balloting group, there were 23 affirmative votes, 15 negative votes, and 8 abstentions. This 82% response was only 60% affirmative. Thus the first ballot failed to make the existing draft a standard.

At the Chicago meeting, objections and comments from all voters (both official and unofficial) were reviewed and acted upon. Many valid points were made by the voters, resulting in changes to the draft. Some revisions included changing the F77 prefixes to PXF (e.g. F77WAIT became PFWAIT). Joseph King's request for a "fast exit" was also added.

Fast exit was added back to the draft to gain the `_exit()` functionality contained in POSIX.1. It is required to allow proper recovery from failed calls to any of the `PXFEXEC()` functions within a child process. It seems that recovery means that the child process must be able to exit without flushing buffers. The file buffers of a child process are copies of the parent's. The current draft says that on failure when `PXFEXIT()`, `STOP`, and `END` are executed, the data in the buffers will be written to the file and the child will terminate. So when the parent writes or closes the file, the output buffers will be flushed and data will be duplicated (once from the failed child and once from the parent) in the file.

Most of the objections and comments were resolved in a positive fashion, providing for the possibility of a successful second ballot. With some fast work from the 8 attendees to the POSIX.9 meeting, the revised draft may be recirculated in June for a 30 day period. If all goes

well, the results of the recirculation ballot can be ready for resolution during the July meeting.

The next meeting of the POSIX.9 working group will be July 8–12, 1991 at the Doubletree in Santa Clara, California. The subsequent meeting will be October 21–25, 1991 in Parsippany, NJ.

Report on 1003.12: Protocol Independent Interfaces

Mike Karels <karels@cs.berkeley.edu> reports on the April 15–19, 1991 meeting in Chicago, IL:

Summary

The POSIX.12 (Protocol Independent Interfaces, PII) working group spent the April meeting planning strategy for its new direction and coordinating with other groups. The group will produce a standard encompassing both the BSD sockets and X/Open Transport Interface (XTI). Liaison meetings were held with X/Open representatives, the Name Space/Directory Services group (POSIX.17) and the Real-Time group (POSIX.4). The group discussed language independent specification issues with Paul Rabin.

Report

POSIX.12 (Protocol Independent Interfaces, PII) spent the April meeting adjusting to its new direction and coordinating with other groups. At the last meeting, the group decided to abandon its previous strategy of producing a new Detailed Network Interface (DNI) with the best features of both the socket and X/Open Transport interfaces (XTI). XTI is derived from AT&T's Transport Level Interface (TLI). After reviewing input from users and vendors, the group decided instead to produce a standard including both existing interfaces. In addition, the standard will include the Simple Network Interface (SNI), which would insulate the programmer from lower-level details.

The April meeting included discussions of the changes or additions that were needed for the existing interfaces to become standards. A poll had been sent to several mailing lists and news groups, but few concrete suggestions were received. Most of the suggestions for extensions have come from inside the working group. Suggestions for changes in sockets have come mostly

from the Berkeley representatives, and suggestions for XTI have come mainly from people active in the X/Open technical community.

A fair amount of time was devoted to the proposal for extending XTI option management by Gerhard Kieselmann. The proposal allows much more flexible option management by encoding option values with types and lengths. The encoding is similar to the encoding of ancillary data in the 4.3-Reno send and receive calls. The main point of contention was whether the transport provider should maintain both current settings and default settings to be used for any future connections.

The discussions of extensions to the socket interface was confined to a description of the recent Berkeley changes (4.3-Reno) to the socket interface.

The meeting schedule was nearly filled with coordination meetings with other groups. Petr Janacek of X/Open reported on the status of future XTI specifications. Other than the option management proposal mentioned above, the XPG4 version of XTI has been finalized. It is hoped that the XPG5 version of XTI will be aligned with the POSIX version. At the last meeting, POSIX.12 asked X/Open for editorial assistance in producing a POSIX version of XTI. Petr replied that the budget did not allow for assistance at this time, but that an on-line version of XTI would be made available.

Paul Rabin met with the PII group to discuss issues surrounding POSIX language independent specifications. The working group currently hopes to produce a single language independent specification for DNI; there would be two C language bindings, namely sockets and XTI. This should prevent the necessity of providing two interfaces for languages other than C, but makes the language independent specification more difficult to produce.

The POSIX.12 group also met with members of the Name Space/Directory Services group (POSIX.17) to discuss the DNI dependency on the Directory Services interfaces. There are some problems in this area. The NS/DS group currently intends to provide an interface only to the X.500 directory service, while the PII group assumes an interface that could include other services such as

;login: 16:4

the Internet Domain Name System. The NS/DNS group intends to provide a full-featured low-level interface to the directory service based on the x/Open X.500 API. However, they also plan to include simplified higher-level interfaces to answer needs such as this one.

The final coordination meetings were with members of the Real-Time group. The current Real-Time draft includes an interprocess communication (IPC) facility that many believe is too complex and does not extend gracefully to handle networked systems. Many hoped that the IPC interface could be replaced by the 1003.12 interface, with real-time extensions as necessary. A group is working on a straw-man proposal in time for the July meeting.

Report on POSIX.17 - Name Space/Directory Services

Mark Hazzard <markh@rsvl.unisys.com> reports on the meeting in Chicago, IL:

Summary

The POSIX.17 group is generating a user to directory services API, for example an API to an X.500 Directory User Agent (DUA). We are referring to a network idea of a directory, not the "file which contains file entries" defined in POSIX.1. It is not limited to just the X.500 functionality. We are using XAPIA — x/Open's Directory Services specification (XDS) — as a basis for work. XDS is an object-oriented interface and requires a companion specification for object management (XOM).

XOM is a stand alone specification with general applicability beyond the API to directory services. It will be used by IEEE 1224.1 (X.400 API) and possibly other POSIX groups, and is being standardized by IEEE 1224.

We made significant progress on a third draft of the document in Chicago, with the language independent specification work still to be done. We hope to mock ballot the document sometime after the July working group meeting. POSIX.12 (Protocol Independent Interfaces) and POSIX.17 worked together this week and arrived at a number of scenarios for coordinating the work. POSIX.17 is taking steps to determine if their work

overlaps with the proposed work of certain ISO/SC21 (OSI) working groups.

Status

Commitment within the group remains adequate, but there's more than enough work to go around.

Chris Harding, (from x/Open) our Technical Editor, brought a second draft of the specification to the meeting. We made significant progress towards producing a third draft with emphasis on format cleanup, model, overview sections and test assertions.

The "homework" assignments on Language Independent Specification (LIS) weren't completed and additional work on LIS was put on hold until the outcome of the SEC meeting. There seemed to be some confusion as to the applicability of the LIS requirement for POSIX.17 and other Distributed Services APIs. The SEC reaffirmed the LIS requirement. The LIS work was reassigned to the Technical Editor.

The big debate on generalizing the Object Management API never materialized. (Refer to the three snitch reports on the New Orleans 1991 meeting.) I strongly suspect this was largely due to the absence of Scott "Owls in the bushes" Guthrey at the Chicago meeting.

Requirements from POSIX.12

The group met with POSIX.12 (Protocol Independent Interfaces) to get their requirements for the POSIX.17 API. They expressed the desire (necessity?) to:

- access existing directory services (e.g. DNS) via the POSIX.17 API

- map the existing BSD API (e.g. `gethostbyname`, `getservbyname`, etc.) onto the POSIX.17 API.

We discussed at length how these and other requirements should best be met, and produced three different scenarios describing relationships between the user application, the directory API(s), the directory service(s), and the transport service (accessed via POSIX.12's Simplified Network Interface).

In the first scenario, the transport provider (SNI) would talk directly to all directory services

e.g. DNS, X.500, etc. Each directory service resolver would be accessed through its native interface, of which POSIX.17 would be just another API.

In scenario two, POSIX.17 would be the only API and would be used to access all directory services. To access a non-X.500 DUA, the underlying implementation might have to translate POSIX.17 calls into the appropriate format and invoke the corresponding resolver.

In the final scenario, POSIX.17 would again be the only API, but only one resolver (X.500 DUA) would be used to query a single composite information base (DIB) containing information on all objects (e.g. DNS Resource Records and X.500 Distinguished Names).

In each of the scenarios, impact to the POSIX.17 API will be minimal. However, significant impact is anticipated for the underlying implementation and directory information base.

We discussed the relative merits of each and decided that at some future time a single API, resolver (agent), directory service, and information base just might be the best for POSIX systems. We also recognized that POSIX systems will need to interoperate with non-POSIX systems for the foreseeable future, and that fact won't be lost on implementors.

Live long and prosper! or Extending the life of our standard

The base document defines both the API and the collection of objects managed through the API, called a "package." We believe that packages will be much more dynamic than the API itself, and could be unbundled from the API to give the API greater stability. We asked the Distributed Services Steering Committee (DSSC) to recommend a common solution, as this problem is shared by other networking groups. We expect the DSSC to take this issue up in Santa Clara.

Mock Ballot

We decided to try to mock ballot our document sometime after the July meeting. After reaching agreement on the minimum document content for mock ballot, we assigned actions to

get this work done. We wish to solicit input on requirements and feedback on our LIS and Test Assertion work.

Is SC21 doing APIs too?

With the granting of any IEEE project request (PAR) comes a responsibility to coordinate with other de jure standards bodies, the list of which is included on the PAR itself. In fulfilling this obligation, the group has learned (and dutifully reported to the SEC) that ISO SC21 is considering working on APIs to OSI application level services. This work has a potential to overlap the SC22 supported work being done by IEEE TCOS/POSIX (e.g. POSIX.17, P1224, P1238).

In Closing

The group made good solid progress in Chicago, and our document is beginning to flesh out. We think we understand what's required for test assertions and language independence, and have done several things to make the base document more readable. If we can maintain critical mass within the group, we have a good chance of going to mock ballot yet this year. There's a lot of work to do, so we hope you can make it to Santa Clara in July.

Report on P1224: X.400 API

Steve Trus <trus@osi.ncsl.nist.gov> reports on the April 15-19, 1991 meeting in Chicago, IL:

Introduction

P1224 is the IEEE working group standardizing an application program interface (API) for X.400 and also for a companion, OSI Object Management (OM). The work will result in two documents. Interfaces developed by the X.400 API Association and X/Open have provided the basis for the standards. The X.400 API consists of two parts: an application interface and a gateway interface. Both of these are based on the 1988 CCITT X.400 Series of Recommendations.

The P1224 working group has the following officers:

- Steve Trus, Chairman (NIST)
- Tim Carter, Vice Chairman (IBM)
- Iain Devine, Technical Editor, Secretary (X/Open)

;login: 16:4

The Chicago meeting was very productive for the P1224 working group. We have been gaining momentum over the past three meetings, and are well under way to producing an IEEE standard.

The goal of the group is to have a draft of the X.400 API and the Object Management APIs by the July meeting, and to ballot the documents after the October meeting.

Report

At the Chicago meeting the group continued modifying the base documents to produce the draft API documents for ballot. This work includes:

1. editing the documents to meet the style and format requirements of the IEEE,
2. adding a language independent specification of the interfaces to the documents, and
3. developing the required conformance test assertions.

The language independent specification of the Object Management API is complete, and the technical editor has made most of the required style changes. These changes will be complete and the language independent specification will be incorporated into the document by the July meeting. Work on the style modifications to the X.400 document will also be complete by the July meeting. The X.400 language independent specification should be complete and incorporated at this time.

The group spent most of the week developing the required test methods for the Object Management Specification. A representative of the Test Methods working group (POSIX.3) assisted us with this development. Members of the group agreed to develop test methods for functions assigned to them by the next meeting. This task will need to be completed before the complete ballot of the document.

Balloting Plans

We discussed balloting plans and we would like to begin balloting the Object Management Specification and the X.400 API in October. These ballots would not include the test methods, and balloting cannot complete without them.

We are developing the list of people who will

be invited to ballot these documents, along with the IEEE-formed balloting group. This list will include the X.400 API Association, X/Open Limited, the NIST X.400 Workshop, and the Electronic Mail Association.

PAR Restructuring

The original Project Authorization Request (PAR) for the P1224 group was written when the baseline document contained an X.400 gateway API and the related OSI Object Management specification. Currently, the X.400 API document contains the user agent interfaces and the gateway interfaces. The OSI Object Management specification is contained in a separate document. To accommodate these changes a revised PAR was written at the January meeting for the X.400 API, and a new PAR was written for the OSI Object Management specification. These PARs were approved by the IEEE TCOS SEC at this meeting.

In Closing

P1224 is making good progress. Homework assignments were delegated at the Chicago meeting to be completed by the Santa Clara meeting. The primary focus of the Santa Clara meeting will be to review the Draft X.400 and Object Management APIs, and to continue working on test methods for the interfaces.

Report on X3J16: C + +

Mike Vilot <mjv@objects.mv.com> reports on the March 1991 meeting in Nashua, NH:

Current Status

The ANSI X3J16 committee began its second year of technical meetings. As expected, the work grew more detailed, with the Core Language and Environment working groups being the focus of most of X3J16's work.

March meeting

Digital Equipment hosted the Nashua meeting. The week's major activities focused on understanding the myriad details of the proposed clarifications and changes to the current working document.

X3J16's sub-groups focused on the key topics listed in the goals statement developed at the

March, 1990 meeting. They worked by electronic mail between meetings, and reported their progress.

International Concerns

Steve Carter, of Bellcore, presented the major international concerns.

Due to the concerns expressed at the November meeting about conversion to a Type I (international) X3 process, Steve came prepared with material explaining the implications of the change. To all appearances, the change seems benign to the technical work of the committee. The change would have the positive effect of getting international involvement. It has the potential to delay the development of the standard, due to the need to synchronize U.S. and ISO balloting.

The full X3J16 committee almost decided to vote to adopt the change, but ran out of the quorum necessary to pass the motion on Friday morning.

Editorial

Jonathan Shopiro, of AT&T, presented the Editorial group's work.

The most significant change from the November version was the incorporation of the exception handling proposal. Jonathan also described an editorial change that simplified the treatment of names and name lookup, merging the concepts that had previously been treated under the topics of dominance and name hiding. Martin O'Riordan, of Microsoft, questioned whether this was a purely editorial change, or a change to the language semantics. Martin and others requested time to look over the change before agreeing to it.

As I mentioned last time, the person who volunteered to edit the Rationale document has not been heard from since last summer. Susan Waggoner, of USwest, has taken on that responsibility.

Formal Syntax

James Roskind, an independent consultant, presented the work of the Formal Syntax group.

The bulk of the discussion concerned a proposal by Reg Charney of Program Conversions,

Inc. to rename the non-terminals in the grammar. Although there was much discussion about the virtues of regularizing the naming versus the evils of gratuitous changes, the committee decided, in the end, to adopt the proposal.

Eric Krohn, of Bellcore, presented the syntactic ambiguities involving the newly-adopted throw-expression syntax for exceptions. The discussion clarified the issues, and a final resolution is likely next meeting.

Tom Penello, of Metaware, gave an interesting presentation on the inherent problems with ambiguous grammars. He established the fact that an ambiguous grammar makes the question of a conforming implementation undecidable. He also illustrated that arbitrary rules to resolve grammatical ambiguities has the side-effect of rejecting valid programs.

He then went on to explain the syntactic ambiguities of the template syntax, arising from the conflict over using the ">" symbol as both a relational operator and a template argument list delimiter. Although he proposed a grammar rewrite that solved the problem, he decided not to recommend it on aesthetic grounds.

There seems to be an appreciation within X3J16 as a whole for the technical issues involved in making the grammar correct. There also seems to be a sentiment in favor of letting the semantic rules settle most of the complex issues.

Core Language

Andy Koenig, of AT&T, presented the Core Language group's work.

Document X3J16/91-0005 describes the group's discussion about the linkage of typedef names and anonymous classes. The group decided it was an Environmental issue, and handed it off to the Environment group.

The group discussed objects created under a condition, and resolved to consider those objects governed by an implicit block scope, as if the programmer had explicitly supplied a compound statement. Discussion is summarized in X3J16/91-0021.

Document 91-0019 covers the discussion of lifetimes for temporary objects created by the

compiler. This issue has not reached closure, although the issues were clarified.

Environment

Peter Chapin, of Vermont Technical College, presented the work of the Environment group.

Document X3J16/91-0011 describes the group's discussion about C/C++ compatibility issues. This discussion is continuing.

The group discussed at length the one definition rule — enforcing the rule that a program must have exactly one definition for a given function, even in the presence of multiple inclusions of inline functions and the potential need for the compiler to generate such functions out of line. Document X3J16/91-0024 summarizes the discussion.

There is a proposal to include a section in the standard on required warnings. Laura Yaker, now at Mentor Graphics, presented some ideas of the sorts of things that might be considered as required warnings. The discussion indicated that this is a difficult issue to standardize, since there is so much variation in environments and implementations. This ongoing discussion is summarized in X3J16/91-0014.

Another ongoing discussion concerns static initialization order for objects in different translation units. Document X3J16/91-0012 summarizes this discussion.

There was some discussion on specifying translation limits in the standard. The discussion seemed to generate more heat than light, and nothing was decided.

Lastly, the linkage of types discussion continues, and is summarized in X3J16/91-0023. Peter described several alternate rules to ensure type-safe linkage of types. A central issue is whether the linkage specification is part of the type. There are interesting arguments for and against this.

Libraries

I presented the Library group's work.

There has been some progress on formulating proposals for submission to X3J16. Aron Insinga of DEC presented his proposal to apply templates

to the definition of the standard string class. His progress has been slowed by the lack of an available implementation supporting templates.

Steve Clamage of TauMetric presented proposed resolutions for almost all of the compatibility issues regarding the C library. Most of the small type insecurities can be handled in a reasonably straightforward manner. There are more substantial issues regarding signals, exceptions and the facilities provided by `longjmp()`.

The `iostreams` proposal continues to receive comment. Many of the UNIX-specific issues have been removed. Addressing these concerns raised an interesting point — should the C++ standard adopt the practice of the C standard, in describing only that certain types exist, or should it describe them as classes and specify their required operations? There was some concern that describing classes would be inefficient, but other concerns that the vague wording without a class description would introduce too much variability among implementations.

Language Extensions

Bjarne Stroustrup, of AT&T, presented the work of the Extensions group.

The group is working through a long list of proposals for changes to the language. A significant number of them came from the Core language group, due to an evaluation of what Andy Koenig calls "language extension by technicality" — where suggestions for changing the wording of the standard would have the effect of changing the meaning of the language.

The current list of language extension proposals includes overloading of the "." operator, a proposal for handling national character set issues with digraphs and new keywords, and the adoption of the "inherited" keyword (as in Apple's implementation).

The largest issue lurking in the Extensions category is the addition of support for run-time type information. There will be much discussion on this topic over the next months.

C Compatibility

Tom Plum, of Plum-Hall, presented the work of the C Compatibility group.

The group continued its investigation of the vocabulary differences between C and C++. They decided to categorize their efforts into groups, covering the language, environment, and library. One likely outcome of their work will be a proposal to adopt the same model of sequence points used by X3J11.

Next events

The next three X3J16 meetings (and their hosts) will be:

- June 17-21, Lund, Sweden (Lund Institute of Technology)
- November 11-15, Toronto, Canada (IBM)
- March 1992, Austin, TX (TI)

Zortech announced plans to host one of the other two 1992 meetings in London.

Membership on an X3 committee is open to

any individual or organization with expertise and material interest in the topic addressed by the committee. The cost for membership is \$250. Contact the chair or vice chair for details.

Chair:

Dmitry Lenkov
HP California Language Lab
19447 Pruneridge Avenue MS 47 LE
Cupertino, CA 95014
(408)447-5279
FAX
(408)447-4924
email *dmitry%hpda@hplabs.hp.com*

Vice Chair:

William M. Miller
Glockenspiel, Ltd
P.O. Box 366
Sudbury, MA 01776-0003
(508)443-5779
email *wmmiller@cup.portal.com*

AUUG

Management Committee

Minutes of meeting 9 December 1991

Held at ACMS, Paddington.

Present: Glenn Huxtable, Michael Tuke, Chris Maltby, Frank Crawford, Scott Merrilees, Peter Karr (from 12:00), Rolf Jester.

Meeting commenced at 10:10 am.

Wael Foda of the AUUG Secretariat (ACMS) was present during the relevant portions of the meeting. Presentations to the committee were made by Bob Kummerfeld of MHS, Joe Watkins and Ellen Gubbins of Symmetry Design, and Hugh Irvine of connect.com.au.

1. Apologies

Apologies were received from Pat Duffy and Andrew Gollan.

2. Minutes of last Meeting (24 September 1991)

Moved (FC/MT) that the minutes be accepted. Carried.

There was no business arising from the minutes other than the items dealt with as part of the agenda.

3. President's Report

Due to Pat Duffy's absence there was no President's report.

4. Secretary's Report

Rolf Jester reported:

4.1 Membership has grown to 663 since the last report in August. All of these are financial. Although there were 637 members on the books at 1/8/91, 231 of those were unfinancial, so we have achieved a significant boost to the effective membership. A break-down of membership statistics is attached.

4.2 Rolf Jester will write a letter to go our with the renewal notices for those members whose subscription is due at the end of this year.

Action: RJ

4.3 It was agreed that ACMS need a network connection urgently to facilitate exchange of mail with members and the committee. Purchase of a modem has already been approved. It was suggested

that Wael Foda obtain a copy of U-ACCESS, a package that will allow ACMS's Macintosh to access the network. MHS sell this package and can install it.

Moved (FC/SM) that expenditure of up to \$500 be approved for this purpose. Carried.

Action: RJ/WF

4.4 Chris Maltby agreed to help Peter Karr get his e-mail connection working.

Action: CM/PK

4.5 Michael Tuke will check whether we have registered "Australian Open Systems Users Group" as a trading name.

Action : MT

Moved (SM/GH) that the Secretary's Report be accepted. Carried.

5. Treasurer's Report

Frank Crawford reported:

5.1 Income & Expense accounts for the financial year to date and Balance sheet as at 9/12/91 are attached. The final contribution from AUUG'91 has yet to be added to these reports (see below).

5.2 \$120,000 of our bank balance has now been transferred to a cash management account, reducing the cheque account balance to just over \$23,000 to cover operational costs.

5.3 Michael Tuke will continue to follow up clarification of AUUG's tax-exempt status. The issue currently rests in the hands of the Australian Tax Office.

Action: MT

5.4 Frank Crawford will prepare a budget by the next meeting.

Action: FC

5.5 ACMS will be asked to start and maintain a register of AUUG assets.

Action: RJ/WF

Moved (RJ/MT) that the Treasurer's Report be accepted. Carried.

6. AUUGN Editor's Report

As the AUUGN Editor was not present, there was no formal report. Frank Crawford reported on behalf of Jagoda Crawford that Vol. 12 issue 4/5 was distributed about a month ago and that issue 6 should be mailed shortly after the end of the year.

7. Summer Conferences

- 7.1 Glenn Huxtable tabled a list of the planned AUUG Summer Conferences in Perth, Adelaide, Melbourne, Hobart, Sydney, Canberra, Brisbane and Darwin, with contact details of their respective organisers. See attached.

The following two motions were proposed (GH/MT) and carried.

- 7.2 AUUG will provide each organiser with an initial cash float, not to exceed \$500, for which the organiser must keep accounts. AUUG will also provide additional support to underwrite each Summer Conference up to \$5000 to cover air-fares and up-front conference costs, with the expectation of recovering most if not all of that figure.
- 7.3 AUUG will open a separate "Summer Conference" bank account. Organisers will be given triplicate deposit books and Bankcard transaction stationery. ACMS will account for transactions on this account and AUUG will pay bills authorised by the organisers.

Action: FC

- 7.4 Glenn Huxtable will request feed-back forms to be completed by all Summer Conference participants so that we can ensure that we are addressing members' needs.

Action: GH

8. Network

- 8.1 It was agreed that as a principle, AUUG will seek to promote easy access by members to the network, that we should favourably consider proposals to that end from any interested party, and that we would consider some form of financial support such as loans if appropriate.
- 8.2 Bob Kummerfeld of MHS outlined a possible service that MHS could provide. It would be a message transfer service to the Internet, initially in Sydney and possibly in Melbourne. Charges would be of the order of \$8 per hour connect time, and there could be a discount for AUUG members. MHS could also offer an installation and support service for users who require that.

A service in Melbourne would require purchase of hardware, modems, software and lines. Some support from AUUG for this could help get it started.

The committee generally supported the idea, and asked Bob Kummerfeld to submit a detailed proposal before the next meeting.

- 8.3 Glenn Huxtable agreed to talk to Dialix in Perth about a similar service there.

Action: GH

- 8.4 Regarding AARNET, it was agreed that we would offer a \$100 discount on next year's AARNET registration fee to those organisations who joined only in the last quarter of this year. Chris Maltby will advise Wael Foda of which organisations are affected.

Action: CM

8.5 Hugh Irvine of connect.com.au presented an outline proposal for a commercial Internet service. The committee supported the idea and asked for a detailed proposal by the next meeting to cover

- discounts available to AUUG members,
- what publicity or other support AUUG should give.

9. Publicity matters.

9.1 Ellen Gubbins provided sets of press clippings showing the result of recent PR activity aimed at raising AUUG's profile.

9.2 Ellen Gubbins provided a design and quotation for the AUUG membership card. The cost of 2000 cards to be delivered in January is \$1071.

Moved (PK/FC) that we approve this expenditure. Carried.

9.3 Ellen Gubbins presented two articles for "Open Forum", our future 6-monthly publication for the general commercial/managerial user. It was decided that Ellen would request another article of a slightly more technical nature from MHS.

9.4 Since no quote for this publication has formally been received or accepted by the committee members present, Rolf Jester will check to see what Pat Duffy had agreed to and obtain a copy of Symmetry's quote. He will then ask for a break-down of the cost into Editorial/writing and print/production costs so that we can obtain an alternative quotation.

Action: RJ

9.5 A press release on the AUUG'92 Call for Papers was submitted by Symmetry and approved with some amendments.

10. Symmetry proposal

10.1 Joe Watkins and Ellen Gubbins presented an outline proposal for a comprehensive "Management Service" aimed primarily at marketing AUUG. The committee agreed to study the proposal and decide on further action, if any, at the next meeting.

10.2 Symmetry's presentation stimulated comments from all present. Some of those comments are recorded below for further discussion. Note that these point were not necessarily agreed.

- * AUUG is at a cross-roads with respect to aims and services.
- * We do need a comprehensive business plan to achieve our agreed goals.
- * The costs envisaged by Symmetry are too high.
- * We could use the existing infrastructure of ACMS and add a part-time marketing person.

- * We should aim at wider coverage of UNIX users.
- * We do need to raise AUUG's profile further.
- * We may need cross-organisational links - AUAOS, UIAMG, OSF etc.
- * We may need to advertise more.
- * We should support local Chapters.
- * We could run local ad hoc seminars when speakers and opportunities present themselves, to serve the local user communities.
- * We must never lose sight of the needs of the technically oriented membership.
- * Our focus should be on member benefits and services, not on numbers and revenue.
- * We should concentrate on doing better what we do now conferences, AUUGN etc. - and maybe pay an publishing organisation to produce AUUGN.

10.3 Peter Karr will arrange a survey of all members and other AUUG delegates to obtain formal feedback on AUUG'91 and on members' expectations.

Action: PK

10.4 Each committee member present agreed to write a brief statement on key AUUG aims and member benefits, and circulate these by e-mail for discussion at the next meeting.

Action: All

11. AUUG'91

11.1 Wael Foda presented the final report of the AUUG'91 conference and exhibition - see attached*. The event was an over-all success, with 499 conference registrations. The exhibition was significantly larger than ever before, 1902 square metres of space having been sold (vs 1131 last year), to 66 exhibitors (vs 40 in 1990). A total of 4463 visitors passed through the show, over 3.5 times last year's count.

11.2 The contribution from the exhibition was \$99,572, well up on last year's \$60,227. However, due to significantly increased Conference costs, the net contribution to AUUG from these events was down to \$15,787, plus another \$12,625 yet to be collected. After deducting costs incurred directly by AUUG, particularly for the AUUG stand at the exhibition, the result was a rough break-even. By contrast, previous years' events have produced significant contributions to the organisation, although the amount of that net contribution has been declining.

12. AUUG'92

12.1 The plan for AUUG'92 will address the criticisms made of AUUG'91, especially the concern about the blatant "commercialism" or attempts at sales pitches in some of the vendors' presentations.

* Report too large to include here. Anyone interested should contact the Secretary.

12.2 In the light of the AUUG'91 results, we must plan to reduce the expenditure for AUUG'92. Wael Foda will present a budget to the next meeting.
Action : WF

12.3 Frank Crawford agreed to contact Stephen Prince to verify that Stephen has accepted the AUUG'92 programme responsibility.
Action: FC

12.4 The primary exhibition space is sold at this stage.

13. Chapters

13.1 Rolf Jester indicated that Ursula Purnell-Webb, former DECUS Board member, and responsible for DECUS's Local User Groups, would be available to consult on how to develop effective local chapters. Her travel costs from Canberra would have to be reimbursed. It was agreed that Rolf Jester should make tentative arrangements for her to spend time with us at a future meeting and to inform the committee of details.
Action: RJ

13.2 We agreed that, after such consultation, we would set out objectives for Chapters and delegate the task to a sub-committee which will involve members in the local areas in the process.

14. Other items

14.1 Rolf Jester has received a request from GEC Alstom for use of our membership list for a mailing. This will be referred to ACMS.
Action: RJ

14.2 Rolf Jester has received a request from a member that we start a database of job vacancies as a service to members seeking employment. It was agreed that we would not be able to provide such a service effectively.

15. Next Meetings

Monday 20 January 1992.
Friday 27 March 1992.
Monday 18 May 1992.

Action: RJ

Meeting closed at 5:00 pm.

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 if you are currently really an AUUG member, examine the mailing label of this AUUGN. In the lower right corner you will find information about your current membership status. The first letter is your membership type code, N for regular members, S for students, and I for institutions. Then follows your membership expiration date, in the format exp=MM/YY. The remaining information is for internal use.

Check that your membership isn't about to expire (or worse, hasn't expired already). 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.

AUUG Incorporated

Application for Institutional Membership

Australian UNIX* systems Users' Group.

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

To apply for institutional membership of the AUUG, complete this form, and return it with payment in Australian Dollars, or credit card authorisation, to:

AUUG Membership Secretary
 PO Box 366
 Kensington NSW 2033
 Australia

• 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.

This form is valid only until 31st May, 1992

..... does hereby apply for

- New/Renewal* Institutional Membership of AUUG \$325.00
- International Surface Mail \$ 40.00
- International Air Mail \$120.00

Total remitted

AUD\$ _____

(cheque, money order, credit card)

* Delete one.

I/We agree that this membership will be subject to the rules and by-laws of the AUUG as in force from time to time, and that this membership will run for 12 consecutive months commencing on the first day of the month following that during which this application is processed.

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.

Date: ___ / ___ / ___

Signed: _____

Title: _____

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

For our mailing database - please type or print clearly:

Administrative contact, and formal representative:

Name:

Phone: (bh)

Address:

..... (ah)

.....

Net Address:

.....

.....

Write "Unchanged" if details have not

.....

altered and this is a renewal.

Please charge \$_____ to my/our Bankcard Visa Mastercard.

Account number: _____ . Expiry date: ___ / ___ .

Name on card: _____ Signed: _____

Office use only:

Please complete the other side.

Chq: bank _____ bsb _____ - a/c _____ # _____

Date: ___ / ___ / ___ \$ _____ CC type ___ V# _____

Who: _____ Member# _____

Please send newsletters to the following addresses:

Name: Phone: (bh)
Address: (ah)
.....
..... Net Address:
.....
.....

Name: Phone: (bh)
Address: (ah)
.....
..... Net Address:
.....
.....

Write "unchanged" if this is a renewal, and details are not to be altered.

Please indicate which Unix licences you hold, and include copies of the title and signature pages of each, if these have not been sent previously.

Note: Recent licences usually revoke earlier ones, please indicate only licences which are current, and indicate any which have been revoked since your last membership form was submitted.

Note: Most binary licensees will have a System III or System V (of one variant or another) binary licence, even if the system supplied by your vendor is based upon V7 or 4BSD. There is no such thing as a BSD binary licence, and V7 binary licences were very rare, and expensive.

- | | |
|--|--|
| <input type="checkbox"/> System V.3 source | <input type="checkbox"/> System V.3 binary |
| <input type="checkbox"/> System V.2 source | <input type="checkbox"/> System V.2 binary |
| <input type="checkbox"/> System V source | <input type="checkbox"/> System V binary |
| <input type="checkbox"/> System III source | <input type="checkbox"/> System III binary |
| <input type="checkbox"/> 4.2 or 4.3 BSD source | |
| <input type="checkbox"/> 4.1 BSD source | |
| <input type="checkbox"/> V7 source | |
| <input type="checkbox"/> Other (<i>Indicate which</i>) | |

AUUG Incorporated

Application for Ordinary, or Student, Membership

Australian UNIX* systems Users' Group.

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

To apply for membership of the AUUG, complete this form, and return it with payment in Australian Dollars, or credit card authorisation, to:

AUUG Membership Secretary
PO 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.

This form is valid only until 31st May, 1992

I, do hereby apply for

- Renewal/New* Membership of the AUUG \$78.00
- Renewal/New* Student Membership \$45.00 (note certification on other side)
- International Surface Mail \$20.00
- International Air Mail \$60.00 (note local zone rate available)

Total remitted

AUD\$ _____
 (cheque, money order, credit card)

* Delete one.

I agree that this membership will be subject to the rules and by-laws of the AUUG as in force from time to time, and that this membership will run for 12 consecutive months commencing on the first day of the month following that during which this application is processed.

Date: ___ / ___ / ___ Signed: _____

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

For our mailing database - please type or print clearly:

Name: Phone: (bh)
 Address: (ah)

 Net Address:

 Write "Unchanged" if details have not altered and this is a renewal.

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: _____ Member# _____

Student Member Certification *(to be completed by a member of the academic staff)*

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

Title: _____

Signature: _____

AUUG Incorporated

Application for Newsletter Subscription

Australian UNIX* systems Users' Group.

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

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
 PO 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, 1992

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 Australian UNIX* systems Users' Group.

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

If you have changed your mailing address, please complete this form, and return it to:

AUUG Membership Secretary
PO Box 366
Kensington NSW 2033
Australia

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: _____

Mem# _____